

SyncWare News

The journal for technical
applications of T/S
computers

Volume 2 Number 6

Jul.-Aug. '85



Contest Nears Conclusion

Congratulations to the 60+ contestants who submitted over 80 programs to the Big Contest. It turned out to be bigger than we thought it would! We are truly impressed, as are the judges from 3 separate user groups, at the quality of so many works of art. The judging has indeed been tough. It has been so tough that as of this writing, 6/24/85, the finalists have not been indentified, let alone the winners. With so many enthusiastic entries, we feel that (there can be only one winner from each machine) there should be more prizes to give out. We're sure that you agree. We have not decided what or how many, but rest assured there will be more than four. We will announce the winners in the next issue. Thank YOU for your support.

Timex Disk Drives

The Timex 2068 disk drive has been submitted to the FCC for approval. We won't know how long it will be before these devices are available until they are actually legalized. The operating system is fully implemented and the disk is a 3" (not 3 1/2") drive. We will have more details in the next issue. It comes with two RS 232 interfaces, a Spectrum emulator and 16K of Ram/Rom in the interface with its own CPU. The ROSCS line (cartridge chip select) is not present at the expansion port.

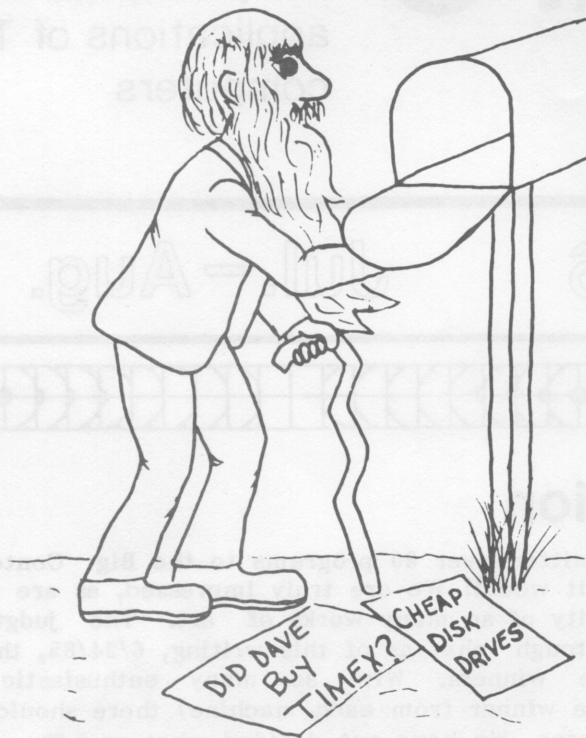
Syntax Error

We have received several letters asking about the status of the publication, Syntax. Although it may be no surprise to many of you, Kirt Olsen of Syntax has stated that he will not take on any new subscribers. He has also said that he would fulfill his obligations. This remains to be seen. If you would like to correspond with Kirt, then drop him a SASE at, The Harvard Group, RD 2, Box 457, Harvard, MA 01451.

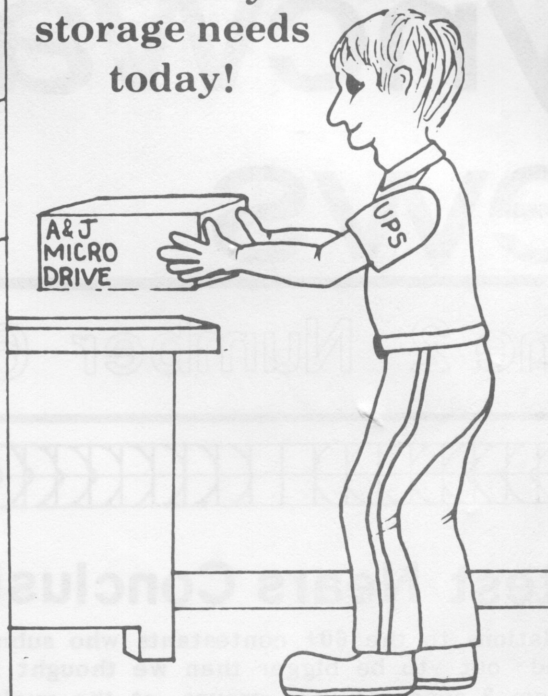
The QL Comes of Age

The QL is out of the FCC (after one year) and is available by mail order and through American Express. We have more details inside. Presently, the only Sinclair peripherals that will be available for it are a near letter quality printer and a hi-res monitor. Jerry Chamkis of Aerco is planning a 512K byte ram board (fits inside) and Ray Kingsley is tinkering with the idea of Hot QL, so there will be American support. Several add-ons, such as disk interfaces, are available in England. A Technical Manual is available from Sinclair Research for \$20.00.

**Don't grow old waiting
for all those Timex
rumors to come true.**



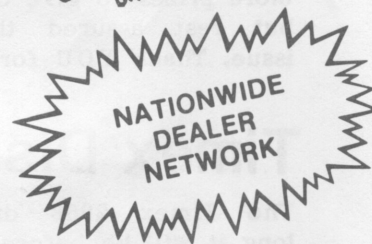
**Let A&J Micro Drive
answer your mass
storage needs
today!**



K. ROLFE
6-24-85
A&J MICRO DRIVE



A&J MICRO DRIVE ANNOUNCES NEW LOWER PRICES



MODEL 2000 STARTER KIT (2068)	\$149.50
includes: Drive, Interface, 5 Micro Wafers, and more	
MODEL 1000 STARTER KIT (1000, 1500, ZX81)	\$149.50
includes: Drive, Interface, 5 Micro Wafers, and more	
MODEL 2068 CENTRONICS KIT	\$24.50
includes: Cable and Instructions	
MODEL 2000 or 1000 OWNER'S MANUAL	\$4.95

MODEL 2000 & 1000 features

- High speed data transfer
- 90 day warranty
- Reliable service
- No cassette hassles
- Lowest cost system available
- Widest dealer support

5' A&J MICRO WAFER	3.50
10' A&J MICRO WAFER	3.50
20' A&J MICRO WAFER	4.00
35' A&J MICRO WAFER	4.00
50' A&J MICRO WAFER	4.50
WAFER ORGANIZER (Folder holds 16 wafers)	5.00
WAFER CADDY (Tray holds 12 wafers)	9.95
WAFER WHEEL (Wheel holds 40 wafers)	18.75

* Centronics kit must be used with
A&J Model 2000 Stringy Floppy

A&J DEALER NETWORK

(No system has better dealer support)

A&J MICRO DRIVE, Sunnyvale, CA	(408) 732-9292
E AUTHOR BROWN, Alexandra, MN	(612) 762-8847
KNIGHTED COMPUTERS, Fulton, NY	(315) 593-8219
GAMES TO LEARN BY, Collinsville, CT	(203) 673-7089
SUNSET ELECTRONICS, San Francisco, CA	(415) 665-8330
ZEBRA SYSTEMS, INC., Woodhaven, NY	(718) 296-2385
CURRY COMPUTER, Glendale, AZ	(602) 978-2902
RMG ENTERPRISES, Oregon City, OR	(503) 655-7484
TEJ COMPUTER PRODUCTS, Los Angeles, CA	
SUMWARE CO., Alden, NY	(716) 631-2204

A&J MICRO DRIVE

1050 E. DUANE AVE., SUITE "I", SUNNYVALE, CA 94086 (408) 732-9292

FOR YOUR SUPPORT

SOFTTAID feed the world tape is available for \$6.25+\$2.00p&h from Susan Zeigler, Software Services, 14307 Ben Brush, San Antonio, TX 78248. It contains 10 of the greatest hits for the Spectrum; 3D Tank Duel, Ant Attack, Starbike, Spellbound, and more. This is for the Band Aid trust fund for Ethiopia.

TEJ Computer Products, 859 N. Virgil Ave., Los Angeles, CA 90029, has Astro-One for the 2068. It is an astronomy program that contains the physical and orbital parameters of the nine planets & their moons. It will convert Julian dates, days of the week, local time to GMT and more. It comes with a 49 page manual for \$19.95

G. Russell Electronics, RD 1 Box 539, Centre Hall, PA 16828, has a Kempston compatible, fully decoded Joystick port that plugs into the cartridge dock. It will work with most Spectrum Games (those supporting the joystick function). It is available for \$19.95.

Integrated Data Systems, 30 Brookmount Rd., Toronto, Ontario, Canada M4L 3N1, has Word Sinc II.5 and Word Font II.5 available for the 1000. A manual is being prepared (large) now that will help in customizing for specific uses. WSII.5-\$30.00 and WFII.5-\$12.50 add \$1.00p&h (Canadian dollars).

Sharp's, Rt. 10, Box 459, Mechanicsville, VA 23111 (804) 730-9697, has a series of strategy games including War in the East, The fall of Rome, Britain Invaded, and more. They are available for \$19.95 each, \$34.50 for two. Send for catalog.

Richard Booth, Sherman Fairchild Lab, Lehigh University, Bethlehem, PA 18015 (215) 861 3951, has Applied Sinclair Subroutine and Programs for the Mathematically Minded, a book that covers 168 pages and a wide range of numerical computation. It contains an annotated listing. The book is available for \$13.00, with tape \$16.00.

J. Keene, 3515 Ingleside Dr. Dallas, TX 75229, has a switchboard that allows you to use the Spectrum Rom and 2068 Rom without having to open your case every time that you want to switch. \$17.00 for board, \$20.00 for Rom, \$35.00 together, ppd.

Pratt Programs, 4820 E. Morris, Wichita, KS 67218-2421, has Broker, a program for 2068 or 16K 1000. This program will help you keep track of the market and mutual funds and tell you when to invest and sell out. \$25.00+ \$3.00p&h.

Robotron Industries, Inc., 7401 W. Maple Terrace, Wauwatosa, WI 53213, has an eeprom programmer kit available. It will program 2716, 2732 and 2764 eeproms and comes with read, write and verify software. It will work with TRS 80 and TS1000. A power supply kit is also available. \$49.95 for the programmer and \$14.95 for the power supply. Write for more information.

Knighted Computers, 707 Highland St., Fulton, NY 13069, (315) 593 8219, has a new catalog available. It contains hardware and business and game software. It includes a \$3.00 off coupon too.

Ray Rash, 2424 SW. 78th St., Oklahoma City, OK 73159, has a catalog with several business and game programs. A biorythm program is also available. Send a SASE.

Fred Nachbaur, Mountain Station Group Box 12, Nelson, BC Canada V1L 5P1, (604) 354 3858, has Memotext Word Processor available for ALL 1000 and 1500 printer interfaces and disk drives or cassette fastload. Specify your own configuration. \$29.95ppd.

The SyncWare News

Thomas B. Woods
Thomas J. Bent
Fred A. Nachbaur

Publisher
Editor
Technical Director

All subscription information, billing and ad artwork should be addressed to:

SyncWare News
PO Box 64
Jefferson, NH 03583
(603) 586 7734

Article submissions, Forum, Support listings, Classified ads and easy questions, should be addressed to:

SyncWare News
9016 Flicker Place
Columbia, MD 21045
(301) 730 7187

Article submissions, issue tapes (when warranted), hardware and related tough questions should be addressed to:

Syncware News
Mountain Station Group Box 12
Nelson, British Columbia, Canada V1L 5P1
(604) 354 3858

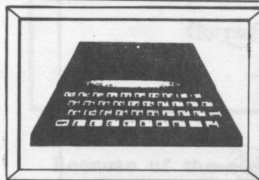
Back issues of SWN are available for \$3.50 each. Volume #1 (for the 1000) is available for \$16.95.

SWN is done entirely on TS computers. Submissions are preferred as word processed text files (so we need not re-invent your article) on good tape. Memotext for the 1000 and 1500 and Mscript for the 2068 are preferred word processors. Documents submitted in Tasword Two and other word processors must be done in 52 column format. If you do not have a word processor, but you still have a good idea, then don't hesitate too work it up and submit it. SWN pays authors about \$40.00 per page for original articles. This amount will vary slightly from issue to issue and is paid when the article is published. Write for more information.

BP asically programming

GET BACK TO BASICS with BASICALLY PROGRAMMING

2528 W. Olive
Fullerton, CA 92633
(714) 738-0666



TS2068 Programs
Multi-Draw 2068.....\$24.95
Softsync Assembler.....\$19.95
Voice Chess.....\$24.95
Personal Accountant.....\$24.95

TS1000 Programs
EZ-HEX.....\$12.95
Programmers Toolkit.....\$14.95
ZX Pro-File (16K to 64K).....\$16.95
ZX Data Finder.....\$14.95
Z-Wryter (16K to 64K).....\$12.95
Profit Plan.....\$12.95
Quest for the Holy Grail
& the Elusive Mr. Big.....\$17.95
3D Black Star.....\$ 9.95
Torpedo Attack.....\$ 9.95
Games Pack (2K).....\$14.95

Please include check or money order. Calif. residents add 6% sales tax. Please add \$2.50 shipping and handling. No COT's - US funds only.

SEND NOW FOR FREE CATALOG!

BP - We have not abandoned basic computer users!

FORUM

Dear Editor, I offer the following change to Jack's Fat Mscript, in order to make it work with the Tasman interface version. Change line 15 to read:

```
15 SAVE "MSCRPT"CODE 36608,8306
```

Lee Bennett, PO Box 67, Lumber City, GA 31549

Dear Editor, I recently bought a 2040 printer that would not work with my 16K Memotech Rampack. I understand that by removing some capacitors from the printer, it will work. If so, will it not work with other Ram configurations? If it is alright to remove them, what is the easiest way?

Jeff Damerst, 307 N. Michael, St. Marys, PA 15857

The easiest way to perform this capectomy is with toenail clippers. Remove the screws from the bottom of your printer. Expose the circuit board and locate the controller chip in the back left corner where the interface wire harness enters. The three caps in question are C4, C5 and C6. They are small tan colored caps. Start clipping. These were initially for FCC requirements. Their absence will not interfere with your computing (just your neighbors TV).

Dear Editor, I noticed that my subscription is about to expire, so I have sent a check to renew. The usefulness of your publishing effort is almost beyond description. Sometimes it is a little too technical for me, but I am amazed at how I grow into it. At any rate, it surely provides new computing and engineering frontiers for me.

Phillp Stevens, 2610 39th St. Des Moines, IA 50310

Dear Editor, I wish Clive would devote more time to creating silicon masterpieces. I think that he could accomplish more by continuing to make it possible for more people to be able to use electrons to move their minds instead of their bodies. Do you know what he is doing these days?

Bob Falk, 39 E. First St., New York, NY 10003

As a matter of fact, he is in the process of selling out his share of Sinclair Research to a Czech magnate. I guess his car is not doing so well. We do wish that he would spend more time on support, but then again we will be here for a long time to come.

Dear Editor, In SWN 2/2, On Loading, by Ed Shaughnessy, after the report code C/190 is given in the Save the Un-saveable, it is possible to List the Un-listable. All Ed needs now is a program for Programming the Un-programmable!

Tom Laffin, Box 133, Hillsboro, NH 03244

Dear Editor, The Forum column article on Spectrum compatability should prove to be useful to most of your readers. You seem almost apologetic in saying that the use of 10K pullup resistors is a "quick and dirty" fix. In my opinion this is the correct solution. Furthermore, since the data lines are at

the expansion connectors, there is no need to open the case to install them.

The keyscan differences of the two machines can cause problems, but most of the software has already been corrected for this. It is the interrupt programs that are most interesting. In Mode 2, the Spectrum does find FF on the bus, but in the 2068, I have found 0F, 2F, 3F, 0E and there may even be others. Some programs don't need an emulator, they need only pullup resistors. One problem that even this won't fix is the speed/TV scan difference. Some programs require a set of calculations to be completed between TV scans. Some Spectrum software may not keep up with the frantic pace of the 2068.

Wes Brzozowski, Sincus News and User Group, 337 Janice St. Endicott, NY 13760 (607) 785 7007

Thanks, I needed that. Wes has a lot more information on this. Send him a SASE and he will fill you in on all the details.

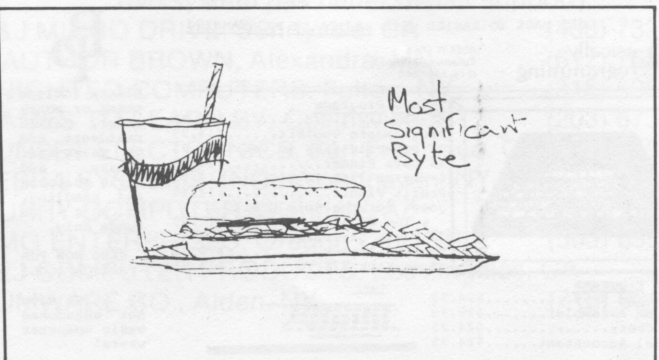
Dear Editor, I plan to try Fred's Load Amp circuit (SWN 2/5) soon. He mentions that one 2068 from GTLB had a single diode bridging the two cascaded ones (CR25 & CR26). I wonder if Timex had come up with the same cure that I did for fried SCLD chips. I believe that that single diode was reversed from the other two. This would give a clamp of no more than -.6V. He also mentions that this computer did not have one of the caps that the others had. This would stand to reason, as a boost in signal strength could be tolerated by the SCLD if it is clamped with the diode to -.6V. I suggest that anybody installing any device to increase the load signal, should add this reverse diode for protection. (Fred agrees. If one of the other diodes goes, you could have problems.) Since the SCLD is a CMOS device, I doubt that it has a comparator in the load circuitry. The Spectrum has schmitt trigger input, and I would say that the 2068 has a CMOS version of this.

As a matter of interest, I expect to have the disk "B" boards by the end of July. All I will need to make the system fly will be Ray's DOS and the DOCS.

John Olinger, 11601 Whidbey Dr., Cumberland, IN 46229

Dear Editor, In the Print Command Compiler, on the last page in SWN 2/5, you have the 18 byte disassembly incorrectly listed. You have the HL register loading from and to the variable "prog" (5C53h). Although it doesn't matter, the correct variable should be "ch_add" (5C5Dh). It may confuse anyone studying the listing though.

William Powers, Maxsoft, 611 Franklin St. Hamilton, OH 45013



TS1500 Fix

In the last issue, we reported a bug in the LOAD command that would put the stack pointer at the address where the bad load occurred. I made a copy of the 1500 ROM (0000h to 1FFFh) at 5000h to 6FFFh. The listings (done with Hot Z) shown here are of the original ROM and the fix for an eeprom.

035D 10F1		DJNZ 0350
035F F1		POP AF
0360 BA		CP D
0361 D2E303		JP NC 03E3
03C3 CDE702	NEW	CALL 02E7
03C5 2A0440		LD HL, (RAMTP)
03C9 2B		DEC HL
03CA 3E3F		LD A, 3F
03CC 44		LD B, H
03CD 4D		LD C, L
03CE 3600		LD (HL), 00
03D0 2B		DEC HL
03D1 BC		CP H
03D2 20FA		JR NZ 03CE
03D4 60		LD H, B
03D5 69		LD L, C
03D6 23		INC HL
03D7 220440		LD (RAMTP), HL
03DA 210020		LD HL, 2000
03DD 35		DEC (HL)
03DE 2002		JR NZ 03E2
03E0 E9		JP (HL)
03E1 00		NOP
03E2 2A0440		LD HL, (RAMTP)
03E5 2B		DEC HL
03E6 363E		LD (HL), 3E

Original TS1500 ROM (with bug)

535D 10F1		DJNZ 5350
535F F1		POP AF
5360 BA		CP D
5361 1863		JR BADT
53C3 CDE702		CALL FAST
53C5 2A0440	BADT	LD HL, (RAMTP)
53C9 54		LD D, H
53CA 5D		LD E, L
53CB 3E3F		LD A, 3F
53CD 2B		DEC HL
53CE 3600	CLER	LD (HL), 00
53D0 2B		DEC HL
53D1 BC		CP H
53D2 20FA		JR NZ CLER
53D4 EB		EX DE, HL
53D5 220440		LD (RAMTP), HL
53D8 210020		LD HL, 2000
53DB 7E		LD A, (HL)
53DC FE01		CP 01
53DE 2005		JR NZ STAK
53E0 E9		JP (HL)
53E1 00		NOP
53E2 00		NOP
53E3 00		NOP
53E4 00		NOP
53E5 2B	STAK	DEC HL

Corrected code for EPROM

Because of the tight space in the 1500, it is necessary to solder on the eeprom and run some jumper wires around on the board. Put some conductive foam (Radio Shack) under your eeprom before you solder on it.

1. After burning your eeprom, bend out pins 20 and 23.

2. Solder wire wrap wire on and connect pins 1, 28, 27 and 26 together.

3. Solder jumper wires on pins 2, 20 and 23.

The 1500 has a 24 pin socket in it, and you are installing a 28 pin chip. Keep this in mind as you read the next few lines.

4. Jumper eeprom pin 2 to pin 21 of the socket.

5. Jumper eeprom pin 20 to pin 20 of the socket.

6. Jumper eeprom pin 23 to pin 18 of the socket.

7. Install the eeprom with pins 1, 2, 28 and 27 hanging over the top of the socket.

These changes can be installed in a 1000. The Command cartridge for the 1500 would then come up running on a 1000 at power up. With Hunter board applications (or other 8 to 16K boards), if you have the instruction LD BCnnnn, where nnnn is any hex number, as the first instruction (at 2000h), then your program will start all by itself. You must take care of the initialization though.

On the 1000 with a 28 pin ROM socket the installation is much easier. Bend out pins 20 and 23. Jumper pin 20 to pin 22 of the socket. Jumper pin 23 to pin 20 of the socket. To avoid soldering on the eeprom, use two 28 pin low profile sockets. Jumper the top socket pins (20 and 23) down to the bottom socket pins (22 and 20). Install the eeprom into the top socket and put the whole thing into your computer. We will have more on this later.

NEW PRODUCTS

GAMESMATE Joystick-Interface: Add speed/convenience to Spectrum programs with Kempston joystick option. Plugs into TS2068 cartridge port. Uses any standard joystick. ROMSWITCH & Spectrum ROM compatible. \$19.95

ROMFIX: Update for ROMSWITCHES produced before 2/85. Only \$3.25

SPECTRUM HARDWARE ADAPTER INTERFACE for TS2068/ROMSWITCH (EMU4): Lets you use Sinclair Spectrum microdrives/other Spectrum hardware. \$35

CHROMA-SOFT: 100% Software generated color for black/white TV for TS1000/1500, ZX81. SEEING IS BELIEVING! Break-through experimental graphics program produces 9 colors by "tricking" the brain. \$14.95

T.A.S. FIX: For Tasman Printer Interface \$24.95 assem./tested

SUMMER SALE PRICES — ends August 31, 1985

ROMSWITCH for TS2068 \$45.; ROMSWITCH + GAMESMATE \$62.95
WINKY BOARD 2000 \$20.95; ZX Spectrum Manual \$12.95
WINKY BOARD 2000 + 007 SPY CASSETTE \$26.95

16K RAM PACS for TS1000, ZX81 \$19.95; For TS1000/1500, ZX81/80-WINKY BOARD 2 \$16.95; WINKY BOARD 2 + ZXLR8 Fastload \$25.95
Shipping included in all prices listed.

FREE CATALOG includes SUMMER SALE, LIQUIDATION ITEMS, SPECTRUM SOFTWARE LIST, NEW PRODUCTS DESCRIPTIONS.

RUSSELL ELECTRONICS

RD 1 Box 539-S, Centre Hall, PA 16828
814-364-1325 MasterCard/Visa 10am-8pm, Check/MO



TI99 Keyboard

So, you got one of those TI99 keyboards from Radio Shack for \$2.95 and still have not wired it up to your 1000 or 2068, right? I was going to write this up a while ago, but I learned that so many different companies made the circuit boards that it would be difficult to give a direct "how to" and cover all of the permutations. The response in favor of the article was overwhelming, so here it is.

It is not necessary to grind off all of board traces and rewire the entire board. It is necessary to make a number of trace cuts though. All of my keyboards have numbered pads on the underside. Since each key is actually a switch, there are two pads for each key. These are denoted as left and right in this project.

A key matrix schematic comes with your keyboard. Mine has the keypads of the numbers 1 through 0 labeled as 1 through 10. A is 23 and V is 38. If yours is numbered the same, then you should have little trouble wiring your keyboard as described herein.

Only the bottom row needs to be cut and moved over one. Fortunately, the TI and ZX matrix are similar. A few of the special keys must also be jumpered around, and the space keys must be linked up again too. Cut the traces above or below the pads specified. Then jumper the necessary pads together.

Cut as follows:

ABOVE	BELOW	
46L	44R	
47L	39L	
33L	38L	
33R	12R	
42R	26R	
41L		
41R		
40R		JUMPER TO
37R		
36R	33L	10L
35R	33R	31L
34R	40R	30R&26R
32R	38R	27R
22R	37R	26R
	36R	25R
	35R	24R
	34R	35L&44R
	39L	40L
	12R	23L
	26R	15R

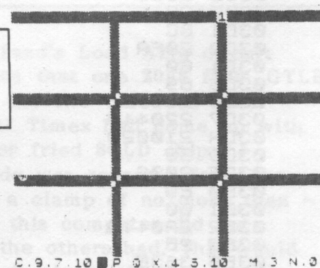
I refer you to Volume 1 of SWN, page 42, and/or the ZX81 or 2068 schematic for the ZX key matrix. You may now directly solder or otherwise connect the ribbon connector of the keyboard to the computer as follows:

KEYBOARD COMPUTER

Pin 1	D5
Pin 2	D3
Pin 3	D4
Pin 4	D7
Pin 5	D8
Pin 7	D1
Pin 8	K0
Pin 9	K4
Pin 10	D2
Pin 11	D6
Pin 13	K1
Pin 14	K2
Pin 15	K3

TIMESCREEN™
ZX81, 16K

Creativity and planning aid:



- 3 screen formats (9-block format shown here)
- BASIC routines to enter and rearrange data
- Calendar
- 10 screens in 16K; 36/32K
- Use with any printer that will accept COPY routine

Cassette and manual, \$9.00 (VA res. add 4% tax), + \$1.00 s&h for one, 50¢ s&h ea. addl. cassette. One 8½x11 Label-Index Card sent with each cass.

HAWC™ Programming

4604 Apple Tree Drive
Alexandria, VA 22310

CLASSIFIED

Do you have something to sell? Let everyone know. Just \$2.75 a line will put it here. 32 column (screen) format, include spaces in your text and send CK. or MO with your listing.

Send to SWN Classified, 9016 Flicker Pl., Columbia, MD 21045

Get cash for that unused board or add-on! SELL IT!!

For Sale. Stringy Floppy, A&J Microdrive for ZX80/ZX81 or TS1000/TS1500. Save programs 30 times faster. Automatic verify with save. Includes manual & wafers, \$125.00. Tom Saine, 2545 Niagara Falls Blvd., Tonawanda, NY 14150

For Sale. Byte Back Modem \$75.00. Includes Software. 16K ram pack FREE! Call 616 323 7822 9-9 or send CK or MO to: Steve Cooley, PO Box 2932, Kalamazoo, MI 49003

PRO/FILE 2068 OWNERS!!
BREAKTHROUGH is the bulletin for keeping your database up to date wit all of the latest upgrades. Issue #1 features a new improved machine code sort routine for fast alphabetizing. Get your copy of BREAKTHROUGH #1 today. Send \$7.95 to: Thomas B. Woods, PO Box 64, Jefferson, NH 03583

Wanted: Memotech RS232 i/f and cable for TS1000. Will pay \$50.00. Anthony Oresteen, 452 Orion, Batavia, IL 60510, (312) 879-5608.

ADDING ON TO GLADSTONE

Many memory expansion boards have been offered for ZX81's and TS1000's. One of the first to break down through the \$100.00 price barrier was the Gladstone 64K RAM pack. It gave you use of RAM memory from locations 8K through 64K. Its only short coming was that you had to ALWAYS use all of the 8K to 64K region. Many printer interfaces, peripherals, etc., are memory mapped to operate in the 8K to 16K region of memory. The circuit used by Gladstone and several other manufacturers, has no provision to disable the RAM pack's control of this region of memory.

One way to get add-ons that are memory mapped to the 8K to 16K region of memory to work with the Gladstone 64K RAM pack is the "hack-and-jumper" method of memory modification. The following provides step-by-step instructions to prevent interference of the Gladstone 64K RAM pack in the 8K to 16K memory region:

1. Open the RAM pack case by removing the three allen head screws and remove the circuit board.
2. Locate IC3, marked U3 on the circuit board, unsolder and remove it.

3. On the component side of the board, cut the circuit trace between pin 3 and pin 10 that has been exposed by the removal of IC3.

4. Replace IC3 and carefully solder it back into the circuit.

5. Cut the trace leading to the card edge connector connection B23 as close to B23 as possible (this is the ROMCS line).

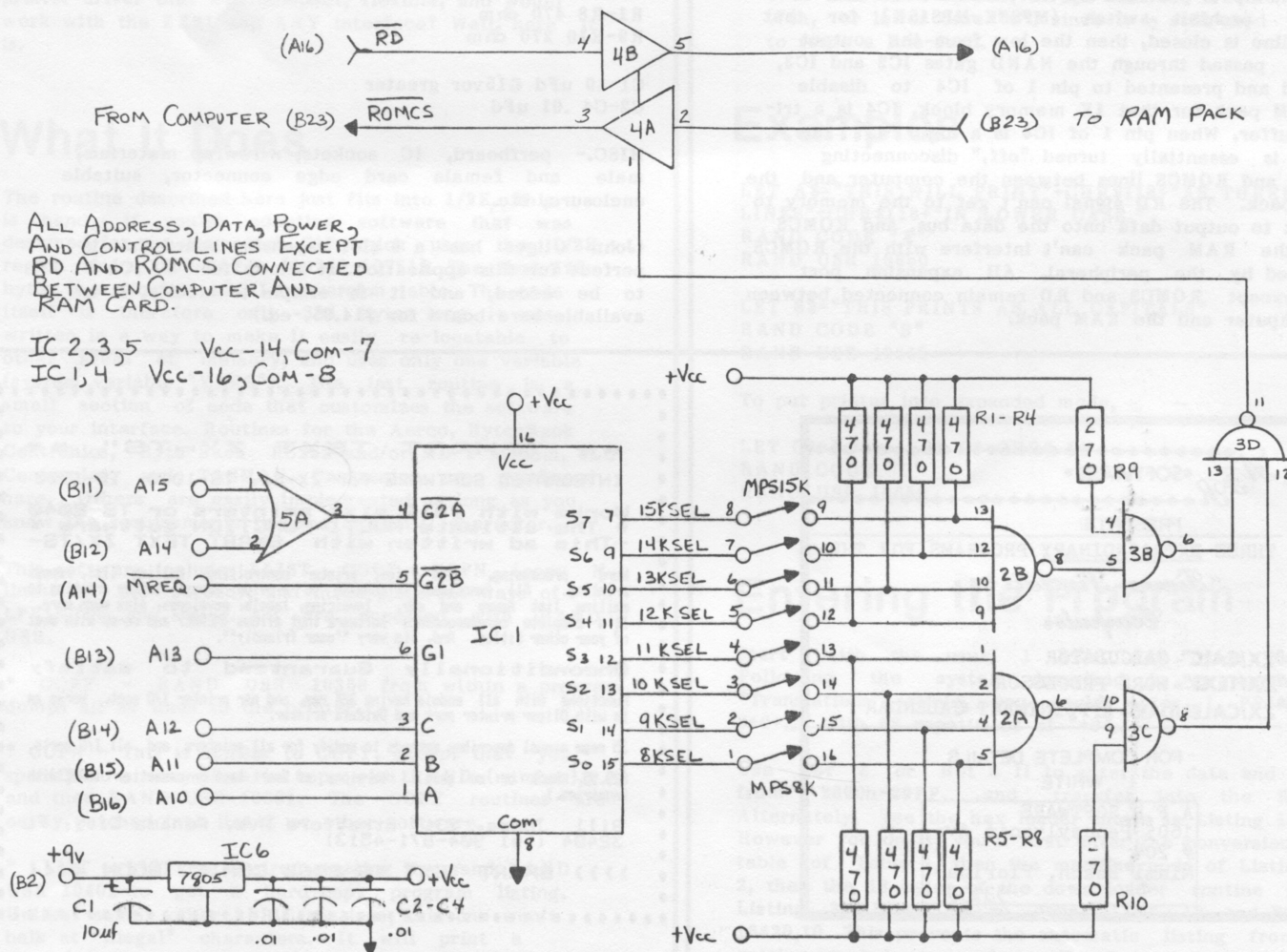
6. Cut the trace between IC3 pin 10 and IC1 pin 6 (IC1 is marked U1).

7. Solder a jumper wire from IC3 pin 3 to IC1 pin 6.

8. Solder a jumper wire from IC1 pin 4 to IC3 pin 10.

9. Check for good solder joints and eliminate any solder bridges.

10. Re-install the circuit board in its case.



The RAM pack should function normally in the 16K to 64K region and allow no memory access or interfere with anything operating in the 8K to 16K region. However, this is a permanent change, and the memory could be damaged physically and electronically with all the hacking and soldering.

A better way to get peripherals to operate in the 8K to 16K region is to construct the circuit shown in figure 1. This is the circuit that Gladstone and other similar RAM pack manufacturers left out. This circuit will allow you to select, by 1K blocks, which parts of memory are enabled or disabled in the 8K to 16K region. The circuit is compatible with any memory that doesn't have provisions to disable all or part of the 8K to 16K region of memory. Also, as long as power is maintained to the memory, a machine code program can be loaded somewhere into the 8K to 16K region of memory, and then the memory prohibit switch(es) controlling that block of memory closed. The program will remain safe in RAM while a peripheral is enabled and operates in the same memory locations. Once use of the peripheral is complete and it is disabled, the memory prohibit switch(es) can be opened and the machine code program will still be in place in RAM, ready to use.

The circuit here uses IC1, a 3-to-8-line decoder/demultiplexer, along with IC5A, an OR gate, to decode the 8K block of memory located from 8192 (8K) to 16384 (16K) into 1K blocks. If A15 (address line 15) and A14 (address line 14, etc.) are both low, MREQ (memory request) is low and A13 is high, then one output line (8KSEL-15KSEL) from IC1 goes low based on inputs provided by A10, A11 and A12. If the Memory prohibit switch (MPS8K-MPS15K) for that output line is closed, then the low from the output line is passed through the NAND gates IC2 and IC3, inverted and presented to pin 1 of IC4 to disable the RAM pack for that 1K memory block. IC4 is a tri-state buffer. When pin 1 of IC4 is a logic "1," the buffer is essentially turned "off," disconnecting the RD and ROMCS lines between the computer and the RAM pack. The RD signal can't get to the memory to allow it to output data onto the data bus, and ROMCS from the RAM pack can't interfere with the ROMCS generated by the peripheral. All expansion port lines except ROMCS and RD remain connected between the computer and the RAM pack.

Construction of the circuit can easily be done using wire wrap and most, if not all of the parts, can be obtained from Radio Shack. Layout of the components isn't critical, but some thought should be given to keep wire runs as short and as neat as possible. Avoid bundling up the wires. All the address, data bus, power and control lines except ROMCS and RD are to be physically connected between the computer, peripherals and RAM pack. ROMCS and RD are connected between the computer and the peripherals, but must go through the tri-state buffer before connecting to the RAM pack. When the wiring and testing is complete, a suitable enclosure should be constructed to protect the circuit and provide RF shielding.

To use the circuit, connect all peripherals except the RAM pack to the computer. Connect the memory adaptor circuit, then connect your RAM pack last. Close the memory prohibit switches for the memory locations where the peripherals are operating, and turn on the power. If all is well, the computer, peripherals and RAM pack will work together in perfect harmony.

PARTS LIST

IC1 74LS138
IC2 74LS20
IC3 74LS00
IC4 74LS367
IC5 74LS32
IC6 7805

R1-R8 470 ohm
R9-R10 270 ohm

C1 10 uFd @15v or greater
C2-C4 .01 uFd

MISC.- perfboard, IC sockets, wirewrap materials, male and female card edge connector, suitable enclosure, etc.,

(John Olinger has a 6 slot expansion board that is perfect for this application. It allows for 6 IC's to be added, and it is simple to modify. It is available bare board for \$14.95.-ed.)

A.F.R. SOFTWARE

PRESENTS:

THREE EXTRAORDINARY PROGRAMS FOR THE

"Timex-Sinclair"

Computer

"ZX/CALC"- CALCULATOR

"ZX/TEXT"- WORD PROCESSOR

"ZX/CALENDAR"- APPOINTMENT CALENDAR

FOR COMPLETE DETAILS
WRITE:

A.F.R. SOFTWARE
1605 Pennsylvania Ave.
204
Miami Beach, Florida
33139

** "SMART TEXT ZX-TS" **

INTEGRATED SOFTWARE for ZX-81, TS-1000, TS-1500

Works with full size printers or TS-2040

* The ultimate MULTI-FUNCTION software *

-This ad written with SMART TEXT ZX/TS-

Word processing, data files, printer controlling, mailing list, repeat printing. All functions INTEGRATED to provide repeat letter printing to mailing list names and adr. Invoicing, labels, envelopes, plus much more. It's complete correspondence software that allows (MERGE) and co-op with most of your other titles. And, its very "user friendly".

Unconditionally Guaranteed to satisfy

Functions with all models having 32K ram, and par printer I/O port. Works as is with Olinger printer port and Okidata printer.

35 page manual describes methods to modify for all printers and all I/O ports.

\$29.95 check or mo gets 3 versions of Smart text on cassette. (SASE with inquiries.)

Bill Jones, 1317 Stratford Ave, Panama City, FL
32404 (tel 904-871-4513)

>>>> SMART TEXT TS-2068 COMING SOON <<<<

UNIVERSAL PRINTER DRIVER

F. Nachbaur

In Volume 1, I ran a series on getting the most out of your ZX/TS with full-size printer and Memotech Centronics interface. Since then, this CIF has become unavailable. Fortunately, several other printer interfaces (both parallel and serial) are available. These include Centronics Parallel IF's by Aerco, Byte-Back, John Oliger, and Tasman, and serial IF's from Byte-Back and possibly others.

Most of these come with "printer driver" software that allow you to LLIST, COPY and LPRINT. At least one (the Aerco) includes an EPROM card that overlays the ROM and allows direct use of the Sinclair commands. The Tasman is the exception (to my knowledge), since it is being imported for the TS2068. (It WILL work with the ZX81/TS1000, however.) Similarly, the Aerco CIF for the TS2068 is identical to the TS1000 version except that it lacks the ROM overlay card. With the appropriate driver software, it too will work with the ZX81 family (and at less cost than the one intended for it).

JLO's driver software (per SQ 1:1) is short and sweet, but lacks LLIST and LPRINT options. Byte-Back's software has LLIST and a sort of LPRINT, but it doesn't allow lower-case and is over 600 bytes long. Wouldn't it be nice if we had a "universal" printer driver that was compact, flexible, and would work with the ZX81 and ANY interface? Well, here it is.

What It Does

The routine described here just fits into 1/2K. This is handy if you're adapting software that was designed for the Memotech CIF which uses the 1/2K region from 2800-29FFh (10240-10751). The first 128 bytes is a Sinclair-ASCII conversion table. The code itself is therefore only 384 bytes long. It was written in a way to make it easily re-locatable to other areas of memory, and uses only one variable (system variable "VERS"). The last routine is a small section of code that customizes the software to your interface. Routines for the Aerco, Byte-Back Centronics, Byte-Back RS232 and/or MD-2 modem, JLO Centronics, and TASMAN Centronics are presented here. Others are easily implemented as long as you know the requirements to send an ASCII character.

This software includes LLIST, COPY, COPN (copy N lines of the screen), and three different ways of LPRINTing any string. All are accessed with RAND USR.

* COPY - RAND USR 10368 from within a program dumps all 24 lines to the printer.

* COPN - This is similar to COPY, except that you specify how many lines to copy with RAND (number), and then RAND USR 10391. The COPY routines are easily patched into Hot-Z or other software.

* LLIST - LIST the desired starting line, and RAND USR 10402 to get a hard-copy program listing. Unlike some other LLIST routines, this one won't balk at "illegal" characters. It will print a tilde (°) instead. It won't stop prematurely due

to an ENTER character (CHR\$ 118) in the middle of a line, so even lines that mess up your screen will LPRINT properly. Like COPY and COPN, LLIST will print "normal" characters as capitals, and inverse video characters as lower-case. (Keeps the tokens looking normal).

* LPRINT - This allows you to print out the contents of any string, simple or arrayed. CHR\$ 118 (ENTER) within the string will force a "newline." To print a string, simply RAND CODE "(string name)", then RAND USR (number). The argument of the USR can be one of three numbers:

10650 (LPRINT 1). This prints inverse video as capitals, and normal video as lower-case. This is what you'd normally use for text, since the inverse characters are more naturally viewed as upper-case (in the style of WSII, Memotext, etc.)

10655 (LPRINT 2). This routine is just like LPRINT 1, except that it prints normal characters as capitals (like COPY and LLIST).

10660 (LPRINT 3). This send the contents of the selected string in ASCII code, without translation. This is handy for sending printer control codes and custom characters. The only code that can't be sent in this fashion is 118 (76h, or lower case "v") since it is still used to force a line-feed.

Examples

```
LET A$="THIS WILL PRINT"+CHR$118+"AS THREE  
LINES"+CHR$118+"IN LOWER CASE."  
RAND CODE "A"  
RAND USR 10650
```

```
LET B$="THIS PRINTS AS ALL CAPITALS"  
RAND CODE "B"  
RAND USR 10655
```

To put printer into expanded mode,

```
LET C$=":" or LET C$=CHR$ 14  
RAND CODE "C"  
RAND USR 10660
```

Entering the Program

Start with the usual 1 REM, 540 bytes long. Following the system described in an earlier "Translations" article, you could enter 1 PRINT 0+0+.... with 68 repetitions of "+0".

Use Hot Z or Hot Z II to enter the data and code from 2800h-29FF, and transfer into the REM. Alternately, use the hex loader shown in Listing 1. However you do it, you'll first enter the conversion table of Table 1, then the machine-code of Listing 2, then the 12 bytes of the down-loader routine of Listing 3. When you're done, LIST 10, and POKE 16420,10. This prevents the automatic listing from getting stuck because of the illegal codes.

Add the BASIC driver of Listing 4 (replacing the hex loader if you used it), and RUN the program to save it to tape. After saving (or subsequently re-loading) the whole business is quickly transferred into its operating region in low memory by the little routine (Listing 3) at 4286h (17030d).

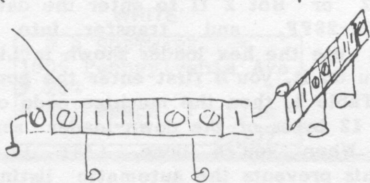
```

00 1 PRINT 0+0+0+0+0+0+0+0+0+0+0+
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+
10 REM HEX LOADER
15 POKE 16513,234
20 FOR I=16514 TO 17041
22 LET AD=I-6274
23 LET H=INT (AD/256)
24 LET L=AD-256*H
25 LET HH=INT (H/16)
26 LET HL=H-16*HH
27 LET LH=INT (L/16)
28 LET LL=L-16*LH
29 LET A$=CHR$( HH+28)+CHR$( H
L+28)+CHR$( LH+28)+CHR$( LL+28)
30 SCROLL
35 PRINT A$,
40 INPUT B$
50 LET A=16*(CODE B$(1)-28)+CO
DE B$(2)-28
60 PRINT B$
70 POKE I,A
80 NEXT I
```

20: 20: 20: 20: 20: 20: 20: 20:
20: 20: 20: 22: 23: 24: 3A: 3F:
28: 29: 3E: 3C: 3D: 2B: 2D: 2A:
2F: 3B: 2C: 2E: 30: 31: 32: 33:
34: 35: 36: 37: 38: 39: 61: 62:
63: 64: 65: 66: 67: 68: 69: 6A:
6B: 6C: 6D: 6E: 6F: 70: 71: 72:
73: 74: 75: 76: 77: 78: 79: 7A:
20: 20: 20: 20: 20: 20: 20: 20:
20: 20: 20: 7C: 25: 26: 60: 21:
7B: 7D: 7E: 5F: 5B: 40: 5D: 7F:
5C: 5E: 27: 20: 30: 31: 32: 33:
34: 35: 36: 37: 38: 39: 41: 42:
43: 44: 45: 46: 47: 48: 49: 4A:
4B: 4C: 4D: 4E: 4F: 50: 51: 52:
53: 54: 55: 56: 57: 58: 59: 5A:

Good
again!

25 compliment



Address	Op Code	Op Name	Comments
2880~1618	COPY	LD D,18 /	>COPY FULL SCREEN
2882 2A0C40	COPD	LD HL,(DFIL)	:Copy D lines
2885 23		INC HL	:Next address
2886 7E		LD A,(HL)	:Get character in A
2887 FE76		CP 76	:End of line?
2889 2805		JR Z 2890	:Yes.
288B CDBD29		CALL NRML	:No. Ship it out,
288E 18F5		JR 2885	and loop back.
2890 CDCD29		CALL LFED	:Send line feed
2893 15		DEC D	:Decrement Line counter,
2894 20EF		JR NZ 2885	and do next line
2896 C9		RET	unless done.
2897 ED4B3240	COPN	LD BC,(SEED)	>COPY N LINES
289B 79		LD A,C	:No. of lines in A.
289C FE19		CP 19 ;	:Over-range?
289E D0		RET NC	:Yes, so return.
289F 57		LD D,A	:No, put it in D,
28A0 18E0		JR COPD	and go to COPD.
28A2 2A0A40	LIST	LD HL,(EPPC)	>LIST PROGRAM
28A5 CDD809		CALL LIAD	:Get line start addr.
28A8 7E	NEXL	LD A,(HL)	:First char. in A.
28A9 FE76		CP 76	:End of program?
28AB C8		RET Z	:Yes, so return.
28AC 46		LD B,(HL)	:No. Get line
28AD 23		INC HL	number
28AE 4E		LD C,(HL)	in BC
28AF 23		INC HL	and increment.
28B0 E5		PUSH HL	:Save the address.
28B1 C5	LNBR	PUSH BC	:Get line no. from BC
28B2 E1		POP HL	into HL.
28B3 11E803	THOU	LD DE,03E8	:Thousands digit
28B6 CDD428		CALL DCML	gets printed,
28B9 116400	HUND	LD DE,0064	then hundreds,
28BC CDD428		CALL DCML	
28BF 110A00	TENS	LD DE,000A	tens,
28C2 CDD428		CALL DCML	
28C5 110100	ONES	LD DE,0001	and finally units
28C8 CDD428		CALL DCML	get shipped out.
28CB AF		XOR A	:Set "space NOT"
28CC 320940		LD (VERS),A	flag, and
28CF CDBD29		CALL NRML	print space.
28D2 180F		JR LLEN	:Continue to LLEN.
28D4 AF	DCML	XOR A	:DECIMAL CONVERSION
28D5 ED52	SUBT	SBC HL,DE	:Subtract modulus.
28D7~3803		JR C BORO	: "Borrow?"
28D9 3C		INC A	:No. Incr. counter,
28DA 18F9		JR SUBT	and subtract again.
28DC 19	BORO	ADD HL,DE	:Yes. Add it back,
28DD C630		ADD A,30 K	convert to ASCII,
28DF CDD829		CALL ASCI	and print it.
28E2 C9		RET	:Return for next digit.
28E3 E1	LLEN	POP HL	:Retrieve addr.,
28E4 5E		LD E,(HL)	and get line
28E5 23		INC HL	length in DE,
28E6 56		LD D,(HL)	point to
28E7 23	NXCH	INC HL	next character, and
28E8 1B		DEC DE	decr. length counter.
28E9 7A	ELN?	LD A,D	:Have we reached
28EA B3		OR E	end of line?
28EB 200A		JR NZ TEXT	:No. It's text of line.
28ED CDCD29		CALL LFED	:Yes, send LF,
28F0 CD460F	BRK?	CALL BRAK	check for BREAK,
28F3 D0		RET NC	and quit if so.
28F4 23		INC HL	:Else incr. add.
28F5 18B1		JR NEXL	and do next line.
28F7 7E	TEXT	LD A,(HL)	:Text char. in A.
28F8 FE7E		CP 7E	:Is is number slug?
28FA 200C		JR NZ NOTN	:No. Go on.
28FC 23		INC HL	:Yes,
28FD 23		INC HL	
28FE 23		INC HL	so
28FF 23		INC HL	
2900 23		INC HL	skip

2901 1B	DEC DE			2981 23	PVAR INC HL	:Next character addr
2902 1B	DEC DE	it,		2982 7A	LD A,D	:Check if counter
2903 1B	DEC DE			2983 B3	OR E	down to zero.
2904 1B	DEC DE	and get		2984 2004	JR NZ GCHR	:No, so go on.
2905 1B	DEC DE			2986 CDCD29	CALL LFED	:Yes, send a LF
2906 18DF	JR NXCH	next character.		2989 C9	RET	and return.
2908 F5	NOTN PUSH AF	:Save character, and		298A 7E	GCHR LD A,(HL)	:Get the character.
2909 E67F	AND 7F	reduce modulus 80h.		298B FE76	CP 76	:Is it an "Enter?"
290B FE40	TKN? CP 40	:Is it a token?		298D 2005	JR NZ PCHR	:No. Print it as-is.
290D 300B	JR NC TOKN	:Yes. Go to TOKN.		298F CDCD29	CALL LFED	:Yes. Send a LF
290F F1	POP AF	:No. Get character,		2992 1803	JR NCHR	and get next char.
2910 CDBD29	CALL NRML	and print it.		2994 CDAD29	PCHR CALL RVRS	:Send it out.
2913 3E01	LD A,01	:Now reset "Space		2997 1B	NCHR DEC DE	:Decrement counter,
2915 320940	LD (VERS),A	NOT" flag,		2998 18E7	JR PVAR	and loop back.
2918 18CD	JR NXCH	next character.		299A 21AD29	LPR1 LD HL,RVRS	: >LPRINT 1
291A~3A0940	TOKN LD A,(VERS)	:It's a token; space		299D 1808	JR 29A7	:Inverse = CAPITALS
291D A7	AND A	already printed?		299F 21BD29	LPR2 LD HL,NRML	: >LPRINT 2
291E 2809	JR Z NLSP	:Yes- don't print again.		29A2 1803	JR 29A7	:Inverse = LOWERCASE
2920 AF	LSPC XOR A	:No.		29A4 21D829	LPR3 LD HL,ASCII	: >LPRINT 3 - ASCII code.
2921 320940	LD (VERS),A	Set "Space NOT,"		29A7 229529	LD (2995),HL	:Set LPRINT CALL,
2924 CDBD29	CALL NRML	print the space,		29AA 18A1	JR LPRT	go to main routine.
2927 1804	JR PRTK	and go on.		29AC 00	NOP	:INV.=CAPS SUBROUTINE
2929 3C	NLSP INC A	:Reset		29AD E5	RVRS PUSH HL	:Save address
292A 320940	LD (VERS),A	"space NOT."		29AE 2628	XLAT LD H,28 C	:Hi byte of conv. table
292D F1	PRTK POP AF	:Get token code,		29B0 CB7F	BIT 7,A	:Inverse?
292E CD7509	CALL TOKA	find ROM address.		29B2 2802	JR Z 29B6	
2931 0A	NTCH LD A,(BC)	:Next char. in token.		29B4~D640	SUB 40	:Yes, subtract 64.
2932 F5	PUSH AF	:Save it, then		29B6 6F	LD L,A	:Low byte of conv. table.
2933 E67F	AND 7F	modulus 80h.		29B7 7E	LD A,(HL)	:Look it up,
2935 FE0F	CP 0F ?	:Illegal "token?"		29B8 E1	POP HL	get address back,
2937 2002	JR NZ OKTK	:No. Valid token.		29B9 CDD829	CALL ASCII	send it out,
2939 3E92	ILGL LD A,92	:Yes. Repl. with a " ~ "		29BC C9	RET	and return.
293B CDBD29	OKTK CALL NRML	Print character.		29BD E5	NRML PUSH HL	:INV.=L.C. SUBROUTINE
293E 03	INC BC	:Incr. address		29BE F5	PUSH AF	:Save Sinc. character.
293F F1	POP AF	and retrieve char.		29BF E67F	AND 7F	:Modulus 128d.
2940 FE80	CP 80	:Was it inverse (last)?		29C1 FE1C	CP 1C 0	:Is it a symbol?
2942 38ED	JR C NTCH	:No. Loop back.		29C3 3805	JR C 29CA	:Yes. Leave unchanged.
2944 AF	TSPC XOR A	:Set "space Not",		29C5 F1	POP AF	:No. Change cases.
2945 320940	LD (VERS),A	and print		29C6 C680	ADD A,80	
2948 CDBD29	CALL NRML	trailing space.		29C8 18E4	JR XLAT	
294B 189A	JR NXCH	:Next character.		29CA F1	POP AF	
294D ED4B3240	LPRT LD BC,(SEED)	: >LPRINT ROUTINE		29CB 18E1	JR XLAT	
2951 79	LD A,C	:Var. char. code in A,		29CD 3E0D	LD A,0D \$	:LINE-FEED SUBROUTINE
2952 C6A0	ADD A,A0	:add 160 (for array		29CF CDD829	CALL ASCII	:Send a CR,
2954 47	LD B,A	var. check). Move to B.		29D2 3E0A	LD A,0A	
2955 ED5B1040	LD DE,(VARS)	:Get VARS start add.		29D4 CDD829	CALL ASCII	and an LF.
2959 EB	NXVR EX DE,HL	:Next variable.		29D7 C9	RET	
295A 7E	LD A,(HL)	:Get var. name.		29D8 E5	ASCII PUSH HL	:PRINT ASCII SUBROUT.
295B B8	CKAR CP B	:Check for string array.		29D9 D5	PUSH DE	for TASMAN interface.
295C 2816	JR Z ARAY	:Yes. Handle as such.		29DA C5	PUSH BC	Save registers,
295E C600	ADD A,80	:No. Add 128 for		29DB F5	PUSH AF	and ASCII char.
2960 2002	JR NZ CKSP	simple string check.		29DC CDE702	CALL TFAS	:Temp. FAST mode.
2962 CF	NFND RST 08H	:Var. not found,		29DF DBBF	STAT IN A,(BF)	:Printer ready?
2963 01	ERROR 2	error back out.		29E1 CB47	BIT 0,A	Loop back
2964 B8	CKSP CP B	:Simple string var.?		29E3 20FA	JR NZ STAT	if not.
2965 2807	JR Z SMPL	:Yes. Handle as such.		29E5 AF	STUP XOR A	:Else set up
2967 C5	PUSH BC	:No. Save search code,		29E6 D3FB	OUT (FB),A	PIO chip
2968 CDF209	CALL NXBV	get next var. addr.,		29E8 3D	DEC A	parameters
296B C1	POP BC	get search code,		29E9 D37B	OUT (7B),A	for "transmit."
296C~18EB	JR NXVR	and start over.		29EB D3FB	OUT (FB),A	
296E 23	SMPL INC HL	:It's a simple		29ED F1	SEND POP AF	Then get char.
296F 5E	LD E,(HL)	string variable		29EE D37B	OUT (7B),A	and ship it out.
2970 23	INC HL	so get start		29F0 3EF7	RSTR LD A,F7	:Now restore
2971 56	LD D,(HL)	address in DE,		29F2 D3FB	OUT (FB),A	PIO chip
2972 180D	JR PVAR	then print it.		29F4 3EFF	LD A,FF	for next time.
2974 23	ARAY INC HL	:It's a string		29F6 D3FB	OUT (FB),A	
2975 5E	LD E,(HL)			29F8 CD0702	CALL MDCK	:Mode check.
2976 23	INC HL	array variable,		29FB C1	POP BC	
2977 56	LD D,(HL)			29FC D1	POP DE	:Restore registers.
2978 23	INC HL	so find out		29FD E1	POP HL	
2979 1B	DEC DE			29FE~C9	RET	
297A 46	LD B,(HL)	how many dims.,		29FF 00	NOP	
297B 23	GTST INC HL /					
297C 23	INC HL	then step addr.				
297D 1B	DEC DE					
297E 1B	DEC DE	and counter				
297F 10FA	DJNZ GTST	till start found.				

ASCII Print Routines For Various Interfaces

BYTE-BACK RS232 INTERFACE AND MD-2

```

29D8~E5      ASCII PUSH HL
29D9 D5      PUSH DE
29DA C5      PUSH BC
29DB F5      PUSH AF
29DC CDE702  CALL TFAS
29DF 3AFF3F  STAT LD A,(3FFF)
29E2 E601    AND 01
29E4 28F9    JR Z STAT
29E6 F1      SEND POP AF
29E7 32FE3F  LD (3FFE),A
29EA CD0702  CALL MDCK
29ED CDF429  CALL DLAY
29F0 C1      POP BC
29F1 D1      POP DE
29F2 E1      POP HL
29F3 C9      RET
29F4 0610    DLAY LD B,10 (
29F6 3EFF    LD A,FF
29F8 3D      DEC A
29F9 28FD    JR NZ 29F8
29FB 10F9    DJNZ 29F6
29FD C9      RET
    
```

DLAY routine may not be necessary with your printer; experiment with different values in 29F5. 120 units=about 1 sec. in SLOW mode. Min. delay is when 29F7=01 an 29F5=01.

J.L.O. CENT. INTERFACE

```

29D8~E5      ASCII PUSH HL
29D9 D5      PUSH DE
29DA C5      PUSH BC
29DB F5      PUSH AF
29DC CDE702  CALL TFAS
29DF 3AFFFF  STAT LD A,(FFFF)
29E2 FE18    CP 18 /
29E4 28F9    JR NZ STAT
29E6 F1      SEND POP AF
29E7 32FFFF  LD (FFFF),A
29EA CD0702  CALL MDCK
29ED C1      POP BC
29EE D1      POP DE
29EF E1      POP HL
29F0 C9      RET
    
```

BYTE-BACK CENT. INTERFACE

```

29D8~E5      ASCII PUSH HL
29D9 D5      PUSH DE
29DA C5      PUSH BC
29DB F5      PUSH AF
29DC CDE702  CALL TFAS
29DF DB1F    STAT IN A,(1F 3)
29E1 CB7F    BIT 7,A
29E3 28FA    JR Z STAT
29E5 F1      SEND POP AF
29E6 D31F    OUT (1F 3),A
29E8 CD0702  CALL MDCK
29EB C1      POP BC
29EC D1      POP DE
29ED E1      POP HL
29EE C9      RET
    
```

AERCO CENT. INTERFACE

```

29D8~E5      ASCII PUSH HL
29D9 D5      PUSH DE
29DA C5      PUSH BC
29DB F5      PUSH AF
29DC CDE702  CALL TFAS
29DF DB7F    STAT IN A,(7F)
29E1 CB67    BIT 4,A
29E3 28FA    JR NZ STAT
29E5 F1      SEND POP AF
29E6 D37F    OUT (7F),A
29E8 00      NOP
29E9 00      NOP
29EA DB7F    RSTR IN A,(7F)
29EC CD0702  CALL MDCK
29EF C1      POP BC
29F0 D1      POP DE
29F1 E1      POP HL
29F2 C9      RET
    
```

Customizing Graphics

You may customize your version to allow LPRINT, COPY and LLIST of graphics characters. As listed, the conversion table sends spaces for these, since not all printers support the block graphics. If yours does, answer "Y" and enter the ASCII codes for each graphic symbol in turn. (Consult SWN Vol. 1 for Gemini 10X, or your printer manual.)

You may want to define the graphics characters to print control codes. (Example: make the graphic on "A" the ESC code 27d, and the graphics on S, D, F, G and H your most-used ESC commands like 10/12 CPI, underline on/off, etc. Be warned though; this may cause strange effects with LLIST/COPY.) SAVE the finished result to tape for future use by answering "Y" on "SAVE?" If you made a mistake, answer "N" and do it over with GOTO 50.

Patching Hot Z

The COPY routines use the D register as a line counter for compatibility with Hot-Z II and other programs that patch into the Sinclair COPY command. To patch the Hot Z II "COPY" command, change the command at 63F9 to CALL 2882. Patching the "COPY cursor to END" is only slightly more involved:

```

5D1D E5      PUSH HL
5D1E CD8628  CALL 2886
5D21 E1      POP HL
5D22 00      NOP
    
```

If you have 32K or more, you can then use Hot Z II to load itself into high memory starting at 8009h (END at B4E5). Then set cursor to 5D1D, END to 63FB, and transfer to 92E1. Save it back to a new tape for future use.

If you have only 16K, then merge HOT Z II with BIGREM, Quit and Save the whole enchilada to your new tape. After re-loading, go back into Hot Z with RAND USR 22528 (cold start) or RAND USR 27273 (warm start; must be from SLOW mode).

Eprom Notes

If you have one of the I/F's that have an EPROM that overlays the ROM and contains its own printer driver, and you wish to use "UPD" instead, you can disable the EPROM. On most devices you can simply disable the EPROM by disconnecting its CHIP SELECT NOT and CHIP ENABLE NOT and tying them to +5V.

To disable the AERCO ROM card, the recommended modification is to cut the trace to pin 11 of the 74LS27 and tie it to +5V instead. (It is presently connected to ground.) Better yet, install a SPDT switch so you can turn the EPROM on and off at will.

The Memotech CIF is the one unit that I know of on which this won't work; it has a PAL chip that does the overlay decoding, and I haven't figured out a way to turn it off and still have the I/O port work.

The Byte-Back Modem and RS232 IF might present a problem with some RAMs since the IF and modem are

memory-mapped at 3FFE-3FFF. If you can't get it to work with RAM enabled in the 8-16K range, there is a one-chip mod to handle it. (If this doesn't appear in this issue, contact me for details.)

Basic Driver Listing 4.

```

1 REM 530 SPACES FOR TABLE, C
  ODE, AND DOWN-LOADER.
2 SAVE "UPD-TAS"
3 PRINT "UPD IS LOADED."
4 USR 10368 = COPY 24 LINES
  AND N, THEN USR 10391 = COPY
  N LINES
  LIST N, THEN USR
  10402 = LIST STARTING AT LI
  NE N
  RAND CODE (VAR. NAME
  )
  THEN USR 10650: INVERSE
  =CAPITALS
  USR 10655: INVERSE=
  LOWERCASE
  USR 10660: SEND AS
  ASCII CODE
40 RAND USR 17030
45 PAUSE 4E4
50 CLS
60 PRINT "INPUT CUSTOM GRAP
  HIC CHARACTERS?"
65 PAUSE 4E4
70 IF INKEY$="N" THEN STOP
80 IF INKEY$<>"Y" THEN GOTO 65
90 PRINT AT 10,0;"CODE CHR$ ",
  "PRESENT NEW ", VAL VAL "
95 SCROLL
100 LET C=0
105 FOR A=1 TO 74
110 IF A=11 THEN LET C=64
115 IF A=11 THEN LET A=64
125 PRINT A+C;TAB 6;CHR$ (A+C),
  ";PEEK (A+16514);TAB 24;"?"
130 INPUT B
140 POKE A+16514,B
150 PRINT AT 21,24;B
155 SCROLL
157 SCROLL
160 NEXT A
165 PAUSE 4E4
167 CLS
170 PRINT "SAVE? Y/N"
175 PAUSE 4E4
180 IF INKEY$="N" THEN STOP
190 IF INKEY$="Y" THEN RUN
200 GOTO 170

```

Relocating UPD

Relocating UPD is simple. Start by changing all CALLs to the new addresses. There are no absolute JPs. The only other commands that need to be changed are at 299A, 299F, 29A4, & 29A7. These load the argument of the CALL at 2994 with the selected routine for LPRINT. Hot Z II's "relocate" command will do this all for you.

The conversion table must start at an address with the low byte = zero (e.g. 2000h, 4100h, etc.) If you relocate the table, then change the command at 29AE to LD H,(new high byte value). This is the only change you'll have to make if you use Hot Z II's "Relocate" command.

DOWN-LOAD TRANSFER ROUTINE

```

4286~010002 LD BC,0200
4289 110028 LD DE,2800
428C 218240 LD HL,4082
428F EDB0 LDIR
4291 C9 RET
4292 00 NOP

```

If you are using the hex loader, enter the hex values (01, 00, 02, etc.) starting at prompt address 2A04

You also have to change the transfer address in the down-load routine, and modify all BASIC USR call addresses accordingly.

Open Invitation

This printer driver is the result of considerable effort by myself and others to get the most flexible and compact software possible. In the interest of standardization, I cordially invite present and future interface manufacturers to supply this routine with their units, PROVIDED THERE IS NO ADDITIONAL CHARGE TO THE CUSTOMER, aside from a maximum \$5 handling charge. These routines are therefore not copyrighted, though I would appreciate mention of the source.

My thanks go out to Donald Thompson, Bill Fisher, Thomas B. Woods, Aerco, Byte-Back and Memotech for their help and contributions.

BACKGAMMON

Looking for a real challenge? Tired of slow response time? Practicing for a tournament? If so, you'll want to try our Backgammon game. Features: full-screen graphics, break disabled, 100% machine code, clean loading, smart play, top rating from TSUG review, multi-game scoring, cube doubling, simple operation, documentation included. Requires ZX81 or TS1000 w/16K RAM. Available on cassette for only \$10. In a 10 game test against PSION, BLOCAL won every game! Write for our FREE catalog listing 100 programs for TS 1000 and 2068. Please include expir. date on MC/VISA orders.



BLOCAL

SOFTWARE, INC.

P.O. Box 965

Stinson Beach, CA 94970

QuarTerS

FOR TIMEX/SINCLAIR ENTHUSIASTS
PUBLISHED BY WMJ DATA SYSTEMS

A quarterly publication with an emphasis on the use of the BASIC computer language as it relates to the T/S computers. QTS provides: programming tips and articles, For you... up to date T/S products, dealers, catalogs, publications, reviews on hardware and software...

"QuarTerS was really impressive this spring and I would like to say, "Well done." Keep up the great work."

Gary Preston II,
Southern VA Timex
Users Network

One year subscription (4 issues) only \$8.00. Overseas subscription only \$11.00(US\$). Sample copy: \$3.00. Check or money order to:

WMJ DATA SYSTEMS

Dept SN

4 Butterfly Drive

Hauppauge, NY 11788

JOIN THE QTS FAMILY
KEEP THE T/S WORLD ALIVE!

BUILDING A 2068 DATABASE

Mike Trivisonno
1322 Golden Gate Blvd.
Cleveland, OH 44124
216 449 1666

The concept of a Completely Interactive Database System (CIDS) that has been implemented on a micro-computer is interesting in that this type of environment is normally only found on mainframe computers. What I hope to do is cover the various aspects of implementing just such a system in the next few issues.

CIDS was created to make various kinds of data easier to work with, although ease of use has its problems, as we will discover. It may not be super duper fast, but it's fast enough. It may not hold as much data as some programs, but it holds enough. What's more, is that it can do more with the data than other programs of a similar nature. (By the way, there aren't many.) CIDS files aren't totally flexible, but they are flexible. For example, try defining graphic characters with a word processor. You'll find it more challenging than you thought! With CIDS it's much easier. CIDS will meet many of your personal needs, but it isn't going to run a business without some help from you.

These 'problems' are not evident when using the CIDS program. CIDS most important ability is its apparent understanding of English commands. The most significant feature incorporated into CIDS is its ability to expand. You definitely will not find this feature in many programs. Integrated software is totally different. This too will be elucidated on as we progress through the CIDS environment.

CIDS Part 1

Type listing one into your computer. Make sure you type it in exactly as shown. Do not change line numbers, add or delete lines. Take your time and do it right. When your done save the program to tape.

The first thing we must do is initialize the variables that we will use and the dictionary of commands CIDS will use. This is accomplished in line 10. This line calls the initialization routine starting at line 9010 and ending at line 9040 with a RETURN statment.

By having the initialization at the end of the listing, the computer only has to execute it once. You must remember that when your computer encounters a GOTO or a GOSUB, it starts at the first line of the program and checks each and every line number until it finds a match. So, for example, if you have fifty lines of code at the beginning of your program that asks for names, numbers or other data and the core of the program that works on this data well below that; every time you tell the computer to GOTO the main part of the program it must check all those lines you put at the top. The computer has to look for the line using brute force; it doesn't have ESP. If you are looking for a word in a dictionary, do you just automatically perceive the page and line number? I think not; you have to look for it by flipping through the pages. Burying the code in this manner will speed up the program execution. Whole articles can be written on this topic alone, But it is

sufficient for us to realize that least used code goes further down in the program.

CIDS is a long program. This installment deals with just the startup routine. Other sections that will surface shortly are the full screen editor and the ADTO routine, each of which are significant in their own right. Long programs usually have several things to do and so, naturally, need several variables. Be sure to follow along closely.

A list of all the variables used so far can be found at the end of this article. Keep it handy. You will also find an annotated listing. Within the article we will go over in detail the operation of each routine as it appears in the program.

The first line of the initialization routine is 9010. This line is the one that sets up the variables that are necessary for CIDS to operate on a simple level. Essentially, it DIMs a large three dimensional array, and also defines three variables mf, ml and mc to keep track of the number of files (mf), the number of lines allowed in a file (ml) and the number of characters allowed in each line (mc). The next statement sets up a two dimensional array that is the same as the number of lines and characters allowed in a single file. You will use this array to store files that are deleted. That way you wish to have the file back, then it is stored away.

The ninth statement defines the variable g. G keeps track of the file with which you are currently working. The variable f keeps track of the number of files that are currently used.

The next statement DIMs the array d\$. This is be our dictionary of commands that CIDS understands and the line numbers where the routines begin. Also defined are some variables that are used for different things at different times. The string variable a\$ holds the commands that are entered.

Lines 9020, 9025, and 9030 fill up the array d\$ with the various routines that CIDS performs. It is important that these lines are entered correctly or program control will most likely send you to unpredictable places. This area of the program will also tell you the beginning line number of each routine. This handy system turns out to be self documentation.

When I first conceived CIDS, I utilized a menu driven routine in which you typed in the name of the routine you wished to access. Since this was cumbersome, I changed to this approach, which is infinitely easier to use. If you think the adto command should be named edit, just change d\$(7) to "edit5000." Be sure that if you do decide to alter the names of any of the routines that you truncate them to only the first four letters as in d\$(6).

The dictionary is what gives CIDS its ability to expand. If you write a routine to replace an existing one, then change the dictionary entry name and viola, your new routine is implemented. It could POKE values into memory locations or define graphic characters. The possibilities are endless. Remove routines and get more memory for data.

Now, with variables initialized, we RETURN to line 10 statement 2, which sends us to another subroutine at line 1506. Now, if you take a quick look at line 9020

statement 3 you will see:

```
LET d$(17)="help1506"
```

This subroutine (help) displays the command content of d\$. After the commands have been displayed, we RETURN to line 10 statement 3. We then go to line 105 from which we go sub the input routine. The input routine is between lines 200 and 290. This routine is essentially an INPUT and string slice subroutine that can be called from anywhere in the program. Imagine never having to write another input routine! It accepts input (in a\$) from the keyboard, looks for the first space and slices the word out. The sliced word is then stored in the variable w\$(for word\$). Nifty eh?

After we slice out the word, we do some error detecting to make sure that no bad things get through. We then RETURN to line 105. We then call yet another subroutine at line 400. This routine then takes the word sliced out of a\$ and compares it with the dictionary that we set up in lines 9020 through 9030. If a match is found, then we hop out of the loop and go to the subroutine specified by the last four characters of the dictionary entry. When we finish the subroutine, we make a check to see if the first character of a\$ is a comma. If it is, we slice it out and go back to the subroutine that we just came back from. If the first character of a\$ is not a comma we simply RETURN. Since the routine was called from line 105 statement 2, we continue where we left off.

This brings us to the BEEP statment. After that, we go back to the first statement, and the whole process starts all over again. We are all done. Fini!

Now that you have the first part of the CIDS program, it is time to go over the first two commands and the screen color change routine. The first one will clear the screen and the second one will display on the screen the names of the commands that CIDS is currently using. The color adjust routine will change the screen to any color you wish.

Let's start with the simplest routine; the clear screen routine. Look at line 9010 statement 2. You will see:

```
LET d$(4)="clea1506"
```

Clea is the name of the routine. 1506 is the line number it begins execution. Well, if you take a look at line 1506 you will see:

```
1506 CLS: RETURN
```

Now, when you are prompted for input and you say something like, "please clear the screen of all this trash," CIDS will eventually locate the word clear. CIDS will then go through the dictionary and find the word clea. Then CIDS will hop over to the subroutine specified by the last four characters of the dictionary entry. Your screen is cleared.

The next routine is not a command, but instead a routine to change the screen color. It is located at line 300. Whatever x equals will be the screen color.

The next command is the help command. The help

command displays on the screen the commands CIDS is currently using. It can be easily modified to display other important information, but what's important is your decision. You might want to display how many files are taken and how many are available. For example, after NEXT t:, add PRINT "files used "; f;"files free ";mf-f

The rest of the lines will be filled in in future instalments of this article. As you can see, there a many of them. For example, the commands make, find, adto, delete and display will be our next exercise.

- * Imagine a full screen editor that lets you insert and delete characters in BASIC! We will have one of those next issue.
- * Imagine an abort command that will restore a file to its original state AFTER you've messed it up!
- * Imagine adding columns of numbers, taking the totals and putting them ANYWHERE in the database!
- * Imagine sorting the database on any record!
- * Imagine windowing in BASIC!
- * Imagine merging new commands to perform specific tasks!

We will cover these tasks and more in upcoming issues as we fully explore the possibilities and usefulness of 'CIDS Building A Database Program.'

For those of you who would like to get a jump on this series, a tape of the CIDS program is available directly from Mike for \$14.95 ppd.-ed.

LIST OF VARIABLES USED SO FAR IN ORDER OF APPEARANCE.

VARIABLE	USE
o	INK color
mf	maximum number of files allowed.
ml	max. no. of lines allowed per file.
mc	max. no. of characters allowed per line.
fl	flag.
f\$(mf,ml,mc)	3-d array for our data
r\$(ml,mc)	2-d array to store deleted file.
g	number of file in use
f	number of files made
d\$(28,8)	holds commands & the lines where they begin all purpose.
h & i	temporary storage area.
s\$	number of previous file.
prev	command string.
a\$	LENth of a\$
l	word sliced out of a\$.
w\$	loop.
w	PAPER color or x pos.
x	loop.
t	

```
10 GO SUB 9010: GO SUB 1506: G
O TO 105
```

```
105 GO SUB 200: GO SUB 400: BEE
P .025,40: GO TO 105
```

```
200 IF a$(1)="" THEN GO TO 240
210 INPUT "READY>> "; LINE a$:
PRINT ">==>";a$
240 FOR i=1 TO LEN a$
261 IF i=1 THEN RETURN
270 IF w$="thanks" THEN PRINT "
Your welcome."
280 IF w$="" THEN GO TO 200
285 IF LEN w$<4 THEN LET w$=w$+
" "
290 RETURN
300 FOR w=1 TO 21: PRINT PAPER
x);
NEXT w: RETURN
410 FOR x=1 TO 28: IF d$(x, TO
4)=w$( TO 4) THEN GO TO 450
420 NEXT x
430 RETURN
450 LET w=w$( TO 4)
455 GO SUB VAL d$(x,5 TO ): IF
a$(1)="" THEN GO TO 465
```

```
460 RETURN
465 IF a$(1)<>"", THEN GO TO 46
0
```

```
470 LET a$=a$(3 TO ): GO TO 455
```

```
1506 CLS : LET x=3: INK 7: GO SU
B 300: PRINT AT 0,7:"COMMANDS AV
AILABLE": FOR t=1 TO 28: PRINT d
$(t, TO 4): PAPER x;: BEE
P .01,50: NEXT t: INK 7: RETURN
9010 LET o=7: POKE 23609,6: LET
mf=26: LET ml=21: LET mc=32: LET
fl=0: DIM f$(mf,ml,mc): DIM r$(
ml,mc): LET g=0: LET f=0: DIM d$(
28,8): LET h=1: LET i=21: LET s
$="": LET prev=0
9020 LET d$(14)="prev6500": LET
d$(4)="clea1500": LET d$(17)="he
lp1506": LET d$(3)="make1510": L
ET d$(1)="find500": LET d$(5)="n
ext1650": LET d$(6)="disp680": L
ET d$(7)="adto5000": LET d$(8)="k
eys8750": LET d$(9)="sort8600":
LET d$(10)="dele1080": LET d$(16
)="name1200"
9025 LET d$(21)="scra1080": LET
d$(22)="loca500": LET d$(23)="cr
ea1510": LET d$(24)="remo1080":
LET d$(25)="left8300": LET d$(26
)="righ8350": LET d$(27)="cent84
00": LET d$(28)="modi8500"
9030 LET d$(13)="adju7000": LET
d$(2)="eval150": LET d$(15)="cop
y1650": LET d$(19)="rest1760": L
ET d$(18)="save1540": LET d$(20)
="load1250": LET d$(11)="zone800
0": LET d$(12)="move1900": LET a
$=""
9040 POKE 23692,255: PAPER 0: CL
S : RETURN
9995 PRINT "-----";w$( TO 4
);
9996 PRINT ("no file" AND w$="ri
ght")+("no file" AND w$="left")+
("no file" AND w$="cent")+("no f
ile" AND w$="copy")+("no file" AN
D w$="move")+("no file" AND w$="
zone2")+("no file" AND w$="edit1
")+("no file" AND w$="bloc1")+("
no file" AND w$="adto")+("out of
range" AND w$="zone1")+("no fil
e" AND w$="dele")+("no file" AND
w$="display");
9997 PRINT ("see pg.28" AND w$="
zone4")+("no file" AND w$="grap
")+("no previous file" AND w$="pr
ev")+("column?" AND w$="zone")+
("next before find" AND w$="next
")+("out of range" AND w$="disp")
+("operator required" AND w$="ev
al")+("name short" AND w$="find"
)+("name short" AND w$="make");
9998 PRINT " pg.": (18 AND w$="di
sp")+ (25 AND w$="left")+ (27 AND
w$="righ")+ (27 AND w$="cent")+ (1
2 AND w$="make")+ (13 AND w$="fin
d")+ (14 AND w$="adto")+ (16 AND w
$="dele")+ (18 AND w$="display")+
(19 AND w$="next")+ (20 AND w$="p
rev")+ (22 AND w$="move")+ (30 AND
w$="zone")+ (32 AND w$="eval")+
(29 AND w$="sort")
9999 PRINT : RETURN
```

```
GO INITIALIZE DICTIONARY AND
VARIABLES: DISPLAY COMMANDS
GO TO INPUT LOOP.
```

```
GO GET INPUT: SEARCH DICTIONARY
DO IT AGAIN.
IF a$ IS NOT EMPTY THEN SLICE OUT NEXT WORD.
IF a$ IS EMPTY THEN ACCEPT INPUT.
LOOP TO SLICE OUT WORD.
FLAG
SOMETHING CUTE
ERROR TRAP
IF w$(ord) IS TOO SHORT THEN PAD IT OUT.
```

```
DONE
SCREEN COLOR ADJUST
```

```
LOOP TO SEARCH d$(ictionary)
IF A MATCH IS FOUND THEN CONTROL GOES TO 450.
KEEP LOOPING
IF NO MATCH THEN RETURN.
CHOP OFF EXTRA CHARACTERS
GO TO THE SUBROUTINE SPECIFIED BY THE
LAST FOUR CHARACTER OF d$(): CHECK a$ AND
IF NOT EMPTY THEN CONTROL GOES TO LINE 465.
DONE
IF THE FIRST CHARACTER OF a$ IS A COMMA THEN
GO BACK TO THE SUBROUTINE YOU JUST CAME FROM.
SLICE OUT (,) AND SPACE THAT FOLLOWS. GOTO
SAME SUBROUTINE.
LOOP TO DISPLAY COMMANDS CURRENTLY AVAILABLE.
```

INITIALIZE VARIABLES.

INITIALIZE DICTIONARY

DITTO

DITTO

DONE

ERROR. PRINT CURRENT w\$(ORD)
PRINT ERROR

DITTO

DITTO

DONE

EXPANDED 2068 INPUT PROMPTS

By Tom Woods

One day, as I was rummaging through the TS2068 ROM, I discovered that there was more to the INPUT command than what was described in the manual. Digging a little deeper, I found that what's presented here is covered in detail in the SPECTRUM manual. Well, so much for international communications.

The TS2068 manual tells us that the INPUT command lets you insert a prompt at the bottom of the screen. No doubt, you have seen this many times:

```
10 INPUT "ENTER DESIRED VALUE:";X
```

The effect of this line is to print at the bottom of the screen the words-ENTER DESIRED VALUE. The input cursor follows this prompt, and the computer waits for you to type in some number which will go into the variable X.

That's pretty basic stuff. Did you know that instead of putting a prompt between quotation marks, you can also refer to string and numeric variables? Consider these lines:

```
10 LET A$="ENTER DESIRED VALUE:"
20 INPUT (A$);X
```

Here, the effect is the same. The prompt gets printed along with the input cursor. It is done not with quotes, but with parentheses and reference to A\$. This opens up many possibilities. For example, in a game for many players:

```
10 INPUT "HOW MANY PLAYERS?";P
20 DIM N$(P,5): FOR X=1 TO P
30 INPUT ("TYPE THE NAME FOR P
LAYER ";X);N$(X)
40 NEXT X
```

And then, later in the game you could have:

```
100 FOR X=1 TO P
110 INPUT (N$(X);", WHAT'S YOUR
MOVE?");A$
120 NEXT X
```

A variation on this theme might be typing in a long list of numbers or words. You can use this quick and dirty input routine to tell you what the current value for a variable is before entering a new value.

```
1 DIM A(10)
10 FOR X=1 TO 10
20 INPUT ("current value for A
(";X;") IS: ";A(X));TAB 0;"NEW V
ALUE? ";B$
30 IF B$<>" THEN LET A(X)= VA
L B$
40 NEXT X
```

You can, in fact, use AT, TAB, INK, FLASH or any other print parameter in conjunction with the input command. The only rule to follow is that any item which begins with a letter of the alphabet will become the variable which takes on the value you type in UNLESS the item is enclosed with quotes.

Have fun experimenting with this. If you want more information on the INPUT command, try to obtain a copy of the Sinclair Spectrum User's Manual.

TASMAN + QUADRA CHART

David C. Ridge
106 Seventh St.
Toronto, Ontario
Canada M8V 3B4
416 259-8682

Dave has come up with some very useful program modifications to better integrate your system. He welcomes and invites any correspondence about 2068 computers.

This program addition will allow you to make full size printer copies from Quadra-Chart.

1. Make a copy of Quadra-Chart by loading it in your computer and breaking the program with the BREAK and CAP/SHIFT keys. Insert a blank tape in your recorder, type GOTO 9990 and press ENTER. Keep watching your screen for the prompts that your computer will give during the SAVE routine.

2. Clear your memory by typing NEW, and then type in the following listing. Note: It is critical that you use the program line numbers exactly as written.

```
15 LOAD ""CODE
118 IF INKEY$="c" THEN GO SUB 9
100
2395 IF J$="c" THEN GO SUB 9100
5565 IF J$="c" THEN GO SUB 9100
7365 IF h$="c" THEN GO SUB 9100
7940 IF INKEY$="z" THEN GO SUB 9
100
9100 REM PRINTER ROUTINE
9105 LET w$=INKEY$
9110 INPUT "Large or small copy-
(L/S)?";w$
9115 IF w$="l" THEN GO TO 9130
9120 IF w$="s" THEN GO TO 9150
9125 GO TO 9105
9130 POKE 23548,0
9135 RANDOMIZE USR 23296
9140 RETURN
9150 POKE 23548,1
9155 RANDOMIZE USR 23296
9160 RETURN
9991 SAVE "wind"CODE 58000,6912
9993 SAVE "line"CODE 51000,6912
9995 SAVE "tas-shinwa"CODE 23296
,255
```

3. DO NOT RUN this program! Insert another blank tape into your recorder and SAVE this program by typing SAVE "quad-mod", without a line number. Press ENTER.

4. Type NEW, and load that copy of Quadra-Chart that you just made. Break the program again and load the TAS-SHINWA graphics driver. Tas-Shinwa is the driver for the Spirit-80 printer. You will have to load the graphics driver for your own printer from the Tasman cassette software. In such a case, change the POKE and RANDOMIZE statements in the above program to suit your particular software driver.

5. Remove the just made copy of Quadra-Chart from your recorder. Load the modification with the MERGE "quad-mod" command.

6. Insert another blank tape into your recorder and save the whole thing by typing GOTO 9990. Be alert for the screen prompts. This will be your finished copy.

DELPHIC ENTERPRISES

ZX-81 and TS-1000 Users !!

2K and 4K Programmer's Utility EPROMs
for your computer.

EPROMs contain routines for editing and development of Basic programs. Routines are Renumber, Copy, Search, Delete, Merge, REM Kill, Free Space, Unlock, Variables Print, Move, REM Generate, Hex/Decimal Converter, Program Size, NVH Storage, Tape Index. Requires Hunter NVH, RomPak, CAI/O or other auxiliary memory board.

2K EPROM (8 routines)	\$18.50
4K EPROM (15 routines)	\$25.00

Delphic Enterprises
P.O. Box 72205
Corpus Christi, TX 78472

QL SUPER BASIC

The QL Comes of Age

by Tom Woods

In the last issue I gave you a brief overview of the hardware angle to Sinclair's latest computer, the QL. This time, I'll tell you about some of the features available in this computer's built-in Super Basic language.

Data Channeling

Like the TS2068 and the Sinclair Spectrum, all input and output is routed through "channels." The TV display, the keyboard, microdrives, and serial ports are all considered channels. If you hook up additional peripherals, you would define new channels in order to input from or output to them.

By associating these channels with Basic commands, such as, PRINT or INPUT, you can move data and programs from device to device with tremendous ease. Unusual effects can be achieved. For example, you can SAVE to the TV screen, or to the serial port. You can PRINT to the Micro Drives, or INPUT from them.

Besides physical devices, channels can be used to output data to WINDOWS, or sections of the TV screen which act completely independently from any other section of the screen. WINDOWS add an interesting new flavor to the art of programming. They can be used, for instance, to produce many interesting displays. After OPENning a channel to the screen, you use the Basic commands WINDOW, INK, PAPER and BORDER; to define size, location, color and border size of a window. Windows can overlap each other and you can have as many as you want at any given moment.

When PRINTing into a window, you can select different character sizes by using the command

CSIZE. The CURSOR command will put the input cursor at the line, column position desired. Once something is printed in a window, you can SCROLL all of it or part of it up or down. Horizontal scrolling (again all or part, left or right) is done using the PAN command.

In addition to Printing to a window, you can also perform all kinds of graphics tricks using CIRCLE (and ellipse), LINE, ARC, and POINT commands. FILL will color objects with a specified ink color.

Programmability

When I first got my QL, sorting out the 140-odd new commands was troublesome. How in the world was I ever going to learn how to use all these new functions? Well, I'm still wrestling with that question, but the more I use this computer, the more I like it.

From a programmer's standpoint, there are some commands which make the QL a joy to use. AUTO will start the automatic line numbering feature. By attaching two numbers to the Auto command, you can specify the starting line and the increment. Also built-in is RENUM which, as its name implies, rennumbers any or all of a program in memory. Add to this extensive micro drive commands: LRUN (load and run) and MRUN (merge and run) are just two of many. Line numbers can range from 1 to 32767, so you should never feel cramped for space.

If you use a monitor with the QL, you can make use of the computer's Hi-res mode which offers the programmer a particularly nice touch. In this mode the main part of the screen is divided into two separate windows, Program listings will go in one, while output from a program goes in the other. Being able to simultaneously watch a display and view a listing is very handy.

Many of the old familiar Basic commands of the TS1000/2068 are presented in the QL with fancy new

WHILE THEY LAST...
BRAND NEW **TOP** QUALITY
DOUBLE SIDE DOUBLE DENSITY
DISC DRIVES \$99

2/3 HEIGHT 400 KILOBYTE CREAM COLORED
DUAL DRIVE CABINET, TOP QUALITY
HEAVY GAUGE STEEL WITH 5 AMP
SWITCHING POWER CONVERTER **\$99**

We want to say *THANK YOU* for the many months of your patience. Our Disc Interface for the TS2068 has 64K of on-board RAM which may be used as a second bank of system memory or a full-blown CPM system. It controls 4 floppy disc drives of any size- from 3 to 8 inches. 1 or 2 sides, single, double, or quad density (8" DD not yet supported). It also includes a complete RGB interface with separated sync so you can *SEE* the High Resolution Color Graphics that this machine produces. It does not include serial or parallel interfaces . . . you add as many as you like at \$69 for 1 Centronics Type Parallel Interface, P/N CP68 \$99 for dual independent RS-232 Serial Interface, RS68 The Disc Board is called FD68 and costs \$199. Deduct \$10 for the first CP68 or RS68. Add \$3 / item shipping.

We are now taking orders.

AERCO
|||||

ACME ELECTRIC ROBOT CO.

Box 18093, Austin, TX 78760
(512) 451-5874

capabilities. FOR... NEXT... is a good example. With the old For.... NEXT.... command, you could set up a repeating loop to execute program lines many times. Each time the lines are repeated, the control variable of the FOR.... NEXT.... command would be incremented or decremented. The Super Basic FOR.... NEXT.... does the same thing, but in addition to simple incrementing or decrementing, you can specify the exact value of the control variable. In the QL such a line would look like this:

```
FOR X=2,4,1,10,5,15 TO 20
```

Here, each time the loop repeats, the variable X takes on the respective values 2, 4, 1, 10, and 5, then 15, 16, 17, 18, 19, and 20. You can compound a line like this to the extreme:

```
FOR X=30 to 80 step 3,5,1,11 TO 25,5,10 11,32
TO 0 STEP -1
```

This loop counts from 30 to 80 by three's; then it counts by 5 and 1; then 11 to 25 by one's; then 5, 10 and 11; and finally, it counts backwards from 32 to 0 by one's. AMAZING!

String & Variable Handling

Remember how other computers' hackers would put down the ZX/TS machines, because you had to use LET when you defined a variable? Well, the QL doesn't make you do this anymore. It will accept:

```
10 A=120
15 B$="hello"
```

But, that's not all it can do. Unlike any other version of Basic that I've seen, QL Super Basic lets you perform the following:

```
10 NUMBER=120
20 A$=NUMBER          look Ma, no quotes!
30 PRINT A$
```

The result of these lines is a string variable consisting of three characters. They read "120." You can further extend this short program with these:

```
40 PRINT A$+A$
50 NEXT NUMBER=A$ & A$
60 PRINT NEXT_NUMBER
```

Line 40 adds 120 and 120 together and prints 240! Line 50 takes "120" and combines it with itself to produce "120120". This value is then assigned to the numeric variable, NEXT_NUMBER.

All this points to some major differences between Super Basic and ZX/TS Basic. If you try adding two strings using the plus sign, the QL will always try to treat the strings as numbers and add the values together. This is true even if A\$="hello" and B\$="good_bye." The computer will look for numeric variables called hello and good_bye and try to add them together.

If you want to combine strings you use the ampersand (&):

```
10 A$="HI "
20 B$="THERE"
30 C$=A$ & B$
```

This will result in a string variable, C\$, which when printed would read HI THERE.

The names that you give variables can be any length up to 255 characters, but you can't use blank spaces. Instead, you separate words with the underscore. Thus,

```
LET DAY_OF_THE_WEEK$="Tuesday"
```

This feature makes writing understandable Basic programs much easier.

Strings are limited to a length of 32K characters. This is restricting, but not constricting. A new command, INSTR, can test the contents of a string for a match to another string. You could say:

```
10 LET DATA$="Now is the time for all good
men..."
20 LET SEARCH$="time"
30 PRINT SEARCH$ INSTR DATA$
```

The result will be a number (12 in this case) which points to the character in DATA\$ where the contents of SEARCH\$ can be found. If a match is not found, then the INSTR function will return a zero. Being the author of ZX PRO/FILE program, I perked right up when I discovered INSTR, because it is very similar to Pro/File's search routine.

I ran some informal speed tests using INSTR and discovered that this function can search through 16,000 characters in somewhere around 5 seconds. That's not nearly as fast as Pro/File, but still its quite respectable. To have a command like this in Basic is very valuable, indeed.

Programs & Prose

Procedures are the most exciting new functions available. They are second generation GOSUBS which are easy to follow and a joy to implement, because like variables, you give them names. However, you give them English names. To run a procedure you just type the name on a program line. The computer keeps track of everything else: where it's going, where it's coming from, etc. [This is very similar to Pascal language in structure, but with the programming ease of Basic.-ed]

If you have ever written programs which make use of subroutines, I'm sure you can see the similarity. The problem with subroutines, however, is that if you make heavy use of them (e.g., GOSUB to a GOSUB to a GOSUB), it doesn't take long before you're up to your neck in Gosubs and you don't know if you're coming or going.

QL's procedures are inherently easier to keep track of because the name you give the procedure can describe exactly what the procedure is intended to do.

There's another advantage to procedures. Variables used within one can be LOCAL to the procedure itself. In other words, you can use the variable X in several different procedures, as well as in the main body of the program. The computer will keep them all separate even though they have the same name. Gone are the hassles of Basic bugs because you accidentally changed the value of some variable.

After you define your procedures and call upon them using their very natural names, your program will look more like written prose than a program. This is extremely helpful when you look at your listing and try to make sense out of your program. Procedure names will tell you just what a group of lines is supposed to do.

To give you an idea of what a "procedural" QL program looks like, take this hypothetical listing of a program that moves a man through a maze:

```

1 REMark Main body of program is here
10 Generate_a_maze
20 REPEAT maze_wander
30 move_the_man
40 AT line,column: print man$
50 END REPEAT maze_wander
60 REMark everything that follows is
    a procedure
70 REMark
80 REMark
90 REMark
100 DEFINE PROCEDURE Generate_a_maze
.
.   lines to make the maze
.
200 END DEFINE
210 REMark
220 REMark
300 DEFINE PROCEDURE move_the_man
.
.   lines to change position of man
.
350 Test_for_walls
400 END DEFINE
410 REMark
420 REMark
500 DEFINE PROCEDURE Test_for_walls
.
.   lines to check man's position
.   against walls of the maze
.
600 END DEFINE

```

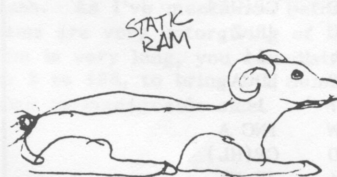
Conclusion

If this piece has whet your appetite for a QL, its not surprising. You could do a lot worse than opting for a QL. From what I've seen of other versions of Basic (Apple II, Commodore 64, ZX, and Microsoft/SANYO), Super Basic beats them all hands down for ease and flexibility of programming.

FLASH!!

Right now, Sinclair is sending out brochures and order forms to everyone who asked to be put on their mailing list. If you do not receive information about the QL from Sinclair, you can request a brochure or even place an order by writing to Sinclair Research, Dept. SWN, 1 Sinclair Plaza, Nashua, NH 03061.

The price for the QL is \$499.00 plus \$12.00 for shipping. Sinclair informs us that Visa and MC are accepted, and that delivery time is 4 to 6 weeks.



BASIL'S COMPENDIUM

Basil Wentworth
1413 Elliston Drive
Bloomington, IN 47401

Deluxe Loader Program

This installment gives you a simple loader program that will make it very easy for you to enter machine code in decimal notation. But first, let's have a look at another "fun" program.

It is assumed by now that you're familiar with the loader program that we have been using, so only the machine code listing is included, in Figure 6-1. And what does this program do?

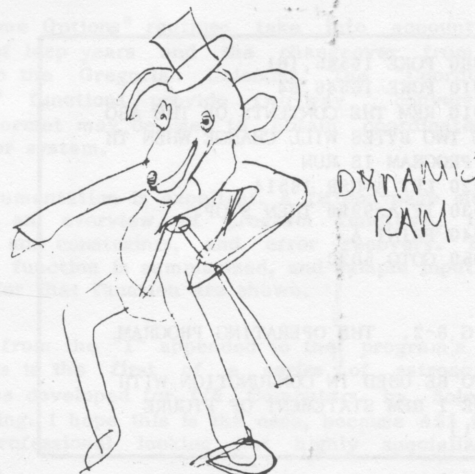
You'll remember, going back to the renumbering program, that you probably would have to change the destinations of all the GOTO and GOSUB instructions, if you wanted the new program to work. Well, this machine code routine will help you, by printing out the numbers of all the lines that include a GOTO or a GOSUB command. Just run the machine code program before you renumber, and you'll know where to look for potential trouble spots. This machine code routine should be used in conjunction with the BASIC mini-program in Figure 6-2.

Or, you can use this program to search for other one-byte entries, such as REM or LPRINT (and LLIST). This last application can be especially useful if you want to avoid inadvertently sending

the computer into an orgy of paper consumption.

And now back to lessons. If your memory is severely limited, you can leave out everything before line 100 (except line 1, of course), once you have learned the procedures they describe.

I almost decided to leave out line 109--not to save the few bytes involved, but because having the line in there might encourage bad habits. Its purpose, as you can probably figure out, is to make sure that the machine code routine ends with RETURN, even if you forget to put it in. I strongly



recommend that you include the RETURN in the program proper, and regard line 109 as a safety net, to help you only in case of emergency. Don't worry about the RETURN being in there twice. The computer, obeying the first one, will never even know that the second one is there.

The danger, of course, is that you'll come to depend on the safety net, and you may find yourself performing some day in another arena (to pursue the metaphor) where they don't have a net. My decision was to include line 109, and then to write my programs as if it wasn't there. This is much like the proverbial worrier who wears both a belt and suspenders. You can do as you like on this point.

```

16514 42 E LD HL,
16515 161 C (16545)
16516 64 RND
16517 62 Y LD A,
16518 236 GOTO 236
16519 35 7 INC HL
16520 190 D CP(HL)
16521 40 C JR Z,
16522 18 > 18
16523 60 W INC A
16524 190 D CP(HL)
16525 40 C JR Z,
16526 14 : 14
16527 62 Y LD A,
16528 117 ? 117
16529 60 W INC A
16530 190 D CP(HL)
16531 32 4 JR NZ,
16532 240 LIST -16
16533 35 7 INC HL
16534 190 D CP(HL)
16535 200 COS RET Z
16536 70 ? LD B,(HL)
16537 35 7 INC HL
16538 78 ? LD C,(HL)
16539 24 / JR
16540 232 CONT -24
16541 34 6 LD(16545),
16542 161 C HL
16543 64 RND
16544 201 TAN RET
16545 161 C (DATA)
16546 64 RND (DATA)

```

FIG. 6-1 THE 1 REM STATEMENT

```

5000 POKE 16545,161
5010 POKE 16546,64
5010 REM THE CONTENTS OF THE ABOVE TWO BYTES WILL CHANGE WHEN THE PROGRAM IS RUN
5020 LET A=USR 16514
5030 IF A>9999 THEN STOP
5040 PRINT A
5050 GOTO 5020

```

FIG 6-2. THE OPERATING PROGRAM

(TO BE USED IN CONJUNCTION WITH THE 1 REM STATEMENT OF FIGURE 6-1)

After you've got the loader program working, SAVE it.

Let me say a few words about the SAVE routine. The lines following 9990 are designed to squeeze out the "excess water" from the program before you SAVE it, to economize on tape and time. (This does not help with the eprom upgrade. The screen will not collapse.-ed.) This technique will work on any program, but it's more impressive on short ones.

Here's the way it works. Get your tape all ready to go (but don't start the recorder), then GOTO 9990. The computer will stop at line 9995. Now make sure that the LOAD link between your recorder and the computer is disconnected (to reduce the danger of feedback), start the recorder going in RECORD mode, and enter CONTINUE into the computer (pressing ENTER, of course).

The tape can then be loaded in the usual manner, and the program will identify itself when loading is complete.

If you want to know why this SAVE-saver works, you'll have to understand that the ZX81 has a parameter called RAMTOP, which is established by bytes 16388/16389. The value at 16389 is tested every time that you clear your screen. If that value is 4Dh or less, then the display file is deleted from memory and is therefore not saved with your program.

Now let's put in that mini-program that we know so well. Change line 100 to read:

```
100 LET P$="6.1.14.0.201."
```

Remember to include the period after each byte, including the last one. Otherwise, the computer won't know where one byte leaves off and the next one begins.

You can proofread the program before you RUN it. And then, for a final check, when you RUN the program, you'll see the relevant byte addresses and their contents lined up for a final inspection before you cross over into USR land.

RUN the program, and then PRINT USR 16514, if the program passes muster at proofreading time. You should get 256, as before.

Now, try the other test program. Change line 100 to:

```
100 LET P$="6.0.14.1.201."
```

and check the result of PRINT USR 16514 (after proofreading, of course).

Just out of curiosity, try a program with no instructions. Except RETURN, of course. ALWAYS include RETURN. Put in LET P\$="201." Then RUN and PRINT USR 16514. So now you know what BC does if you don't give it any instructions. It just takes the value of the starting byte--in this case, 16514.

Now you have seen that the loader not only facilitates proofreading, but it also makes editing easier. Exchanging the values of data bytes 1 and 0 was just a simple matter of editing line 100. No need to POKE, no need to retype the entire program. Correcting a typing error would have been just as easy. This capability really helps when you find

FIG 6-3. THE LOADER PROGRAM

```

1>REM .....
10 PRINT "LIST CODE IN DECIM
AL AS:"
15 PRINT "100 LET P$=""6.1.14.
0.X.X.X.201.""
20 PRINT "REMEMBER TO PUT ""
"" AFTER EACH"
25 PRINT "BYTE, INCLUDING THE
LAST ONE"
30 PRINT "MORE CODE CAN BE A
DDED AS:"
35 PRINT "101,102, ETC. LET P
$=P$+""
40 PRINT "NOW MAKE SURE T
HAT THE 1 REM"
45 PRINT "LINE CONTAINS AT LEA
ST AS MANY"
50 PRINT "NULLS AS YOU HAVE BY
TES IN THE"
55 PRINT "PROGRAM"
60 PRINT " THEN RUN THE PROG
RAM"
100 STOP
109 LET P$=P$+"201."
110 CLS
120 FOR F=0 TO LEN P$

```

```

130 IF P$="" THEN GO TO 220
140 LET C$=""
150 LET C$=C$+P$(1)
160 LET P$=P$(2 TO )
180 IF CODE P$<>27 THEN GO TO 1
50
190 POKE 16514+F,VAL C$
200 LET P$=P$(2 TO )
210 NEXT F
220 FOR M=16514 TO 16513+F
230 PRINT M;" ";PEEK M
240 NEXT M
250 STOP
8000 FOR F=26708 TO 26715
8010 PRINT F,PEEK F
8020 NEXT F
8030 STOP
9990 REM END
9991 LET A=PEEK 16388
9992 LET B=PEEK 16389
9993 POKE 16388,PEEK 16400
9994 POKE 16389,PEEK 16401
9995 STOP
9996 SAVE "LD"
9997 POKE 16388,A
9998 POKE 16389,B
9999 PRINT "DECIMAL LOADER"

```

that you have inadvertently left out a byte. Under the "hunt and POKE" system, you'd have to locate the error, add another null, and then rePOKE the offending area--and possibly the entire program following the error.

It's not such a big deal when the entire program consists of 5 or 6 bytes, but you can imagine what it would be like to retype a few dozen or more bytes (and then proofread them again) just to get rid of one typographical error.

Any error, however small, is very likely to make the program crash. As I've mentioned before, machine code programs are very unforgiving of mistakes! If your program is very long, you can distribute P\$ among lines 2 to 108, to bring the number of bytes per line down to manageable size.

And now we're finally ready to start playing that violin. Well, almost...

CELESTIAL COMPUTING

Astronomical Software 1

Scientific Computing
P. O. Box 5091
Littleton, Colorado 80123
Timex/Sinclair 1000, 1500; \$19.95

AS1 provides functions frequently found only on more expensive software for more expensive computers. Combined into two compact "Time Options" and "Coordinate Options" menus, we see many selections that were treated as separate programs in Eric Burgess' 'Celestial Basic.'

TIME OPTIONS

1. Date to Day Number
2. Date to Julian Day Number
3. Julian Day Number to Date
4. Date to Day of Week
5. UT to LMST
6. Coordinate Options

COORDINATE OPTIONS

1. Oblquity of the Ecliptic
2. Angle Between Two Objects
3. Precesses Coordinates
4. Equatorial<>Horizon
5. Equatorial<>Ecliptic
6. Equatorial<>Galactic
7. Time Options

The monitor screen is divided into three segments for program input, output and prompts. Your input is echo-printed at the top left corner of the screen. Program output is vertically centered. Additional options appear at the bottom of the screen after the current computation is complete.

All date inputs are made with the month in numerical form. A decimal point is required to separate hours and degrees from fractional parts.

The "Time Options" routines take into account the effect of leap years and the changeover from the Julian to the Gregorian Calendar. The "Coordinate Options" functions provide two-way conversions. Either format may be used to obtain coordinates in the other system.

The documentation is excellent. A fifteen-page manual provides an overview of program functions, input formats and constraints, and error recovery. Each program function is summarized, and sample input and output for that function are shown.

I infer from the "1" appended to the program's name that this is the first of a series of astronomy programs developed for T/S Computers by Scientific Computing. I hope this is the case, because AS1 is a very professional looking but highly specialized product.

Reviewed by Duncan Teague

Planet Finder

Michael E. Zerbo
16 Colony Drive
West Sayville, New York 11796
Timex/Sinclair 1000, 1500, \$5.95; 2068, \$9.95

Planet Finder as a title is a misnomer. The program doesn't find planets as does an Ephemeris. It isn't Eric Burgess' Skyplot, a telescopic view of a section of the ecliptic. It also isn't Burgess' Planet Finder, a panoramic view through a picture window at a span of the Earth's horizon.

Planet Finder is a collection of astronomical images and information. It's a cleverly designed, graphically interesting file of solar system data, celestial bodies, and planetary configurations.

Planet Finder loads in two parts. The first program opens with the theme from a classic science fiction movie, produces an animated drawing that reminds me of a local advertisement for water beds, and finally displays a menu of four choices:

1. Load Main Program
2. Review Program Instructions
3. Review Astronomy Terms
4. Terminate Program

Option 2 contains instructions for use of the main program. Option 3 defines astronomical terms ranging from "albedo" to "terrestrial planets." Option 1 loads the main program with these options:

1. Review Planet Information
2. Planets And Their Satellites
3. Facts About The Moon

4. Planetary Orbital Tilts
5. Weight Simulation
6. Questions About The Solar System
7. Terminate Program

Options 1 and 3 remind me of a slide set I've used in my astronomy classes. Each slide pictures one of the major objects in our solar system. Next to each picture is that body's vital statistics. These two menu options produce a screen display which differs from those slides only in regard to the position and quantity of the information.

Option 2 is simply an exhaustive listing of currently known satellites. Option 4 is a diagram with the scale exaggerated to avoid clutter. Option 5 includes a drawing of a double-pan balance. Above the balance calculations compare the weight of an object on Earth with its weight on another planet.

Option 6 is a game of trivial pursuit where the category is always astronomy. The test covers general knowledge, the terrestrial planets, the gas giants, and solar system leftovers (the Sun, the Moon, and Pluto). The four test areas can be taken separately or all together. In the latter instance, a grade is given on each part and on the whole test.

The astronomical data is painstakingly accurate and up to date. It correctly shows Pluto as the smallest planet, Neptune as most distant, and Saturn as the planet with the most moons. This program is lighter than an astronomy text and more fun to use.

The TS/1000 version is less colorful, but it loads itself! Both are a great buy at the prices asked.

Reviewed by Duncan Teague

SyncWare News

CONTENTS

For Your Support	3
Forum	4
Classified	6
Adding on to Gladstone 64K	7
(TS1000) Moore	
Universal Printer Driver, TS1000	9
Nachbaur	
Building a 2068 Database	14
Trivisonno	
Expanded 2068 INPUT Prompts	17
Woods	
Tasman + Quadra Chart	17
Ridge	
QL Superbasic	18
Woods	
Basils Compendium	21
Wentworth	
Astronomical Software 1 REVIEW	23
(TS1000) Teague	

FIRST-CLASS MAIL
U.S. POSTAGE
PAID
Jefferson, NH
Permit No. 1

Editor:
Tom Bent
9016 Flicker Place
Columbia, MD 21045

Technical Director:
Fred Nachbaur
902 Hoover St.
Nelson, BC Canada V1L-4X6

Advertising, Subscriptions:
Thomas B. Woods
P.O. Box 64
Jefferson, NH 03583

Submissions, announcements, letters to the editor, and all other material of editorial interest should be sent to the Maryland address.

Copyright 1985 SyncWare News

Subscribe ! \$ 16.95

1 year (6 issues)

Add \$3 a year in Canada; all other foreign add \$5 per year.